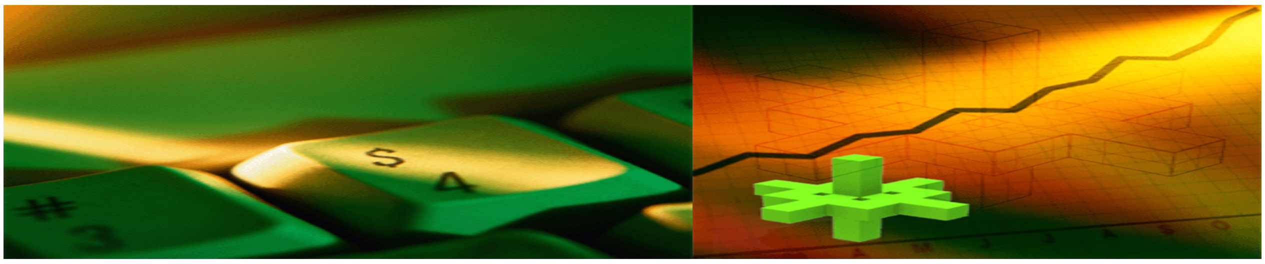


Specification and Verification of Security Policy Design based on Petri Nets

Hejiao Huang

黄荷姣

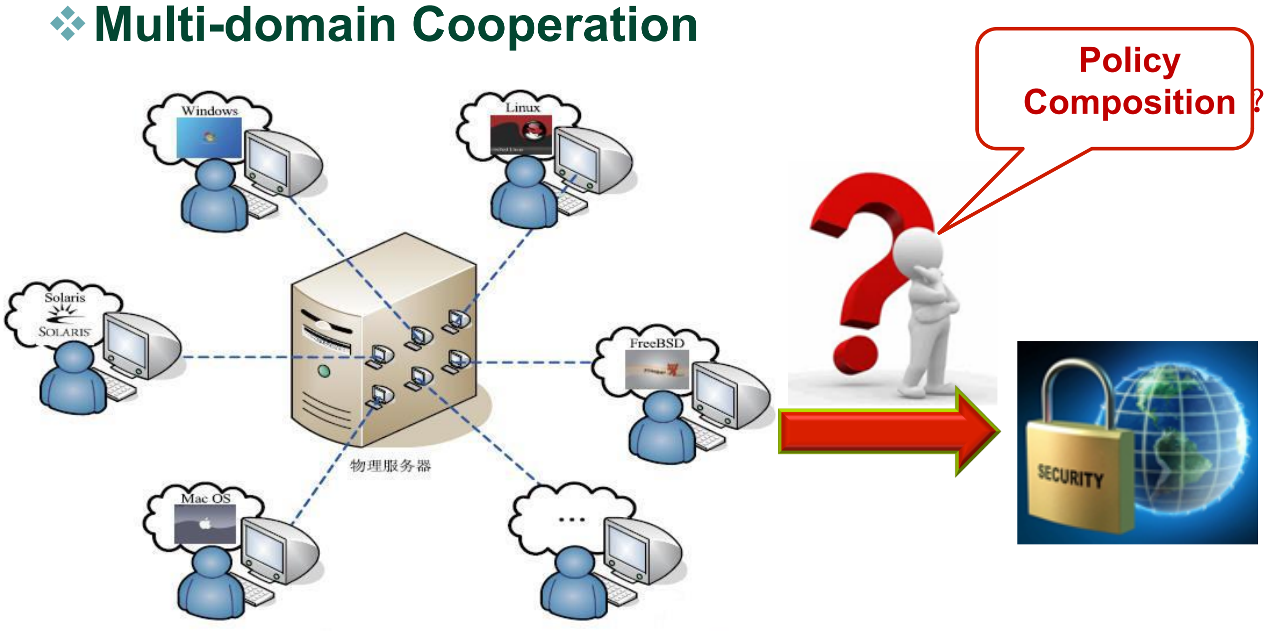
哈尔滨工业大学深圳研究生院
Harbin Institute of Technology Shenzhen Graduate School

Contents

Contents

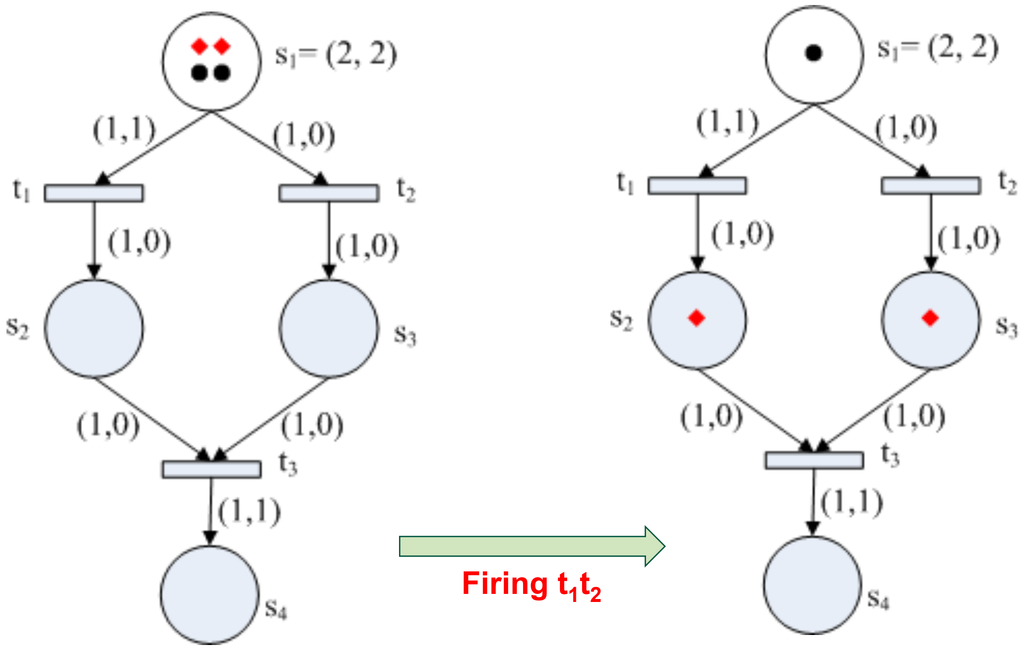
Policy Composition

❖ **Multi-domain Cooperation**

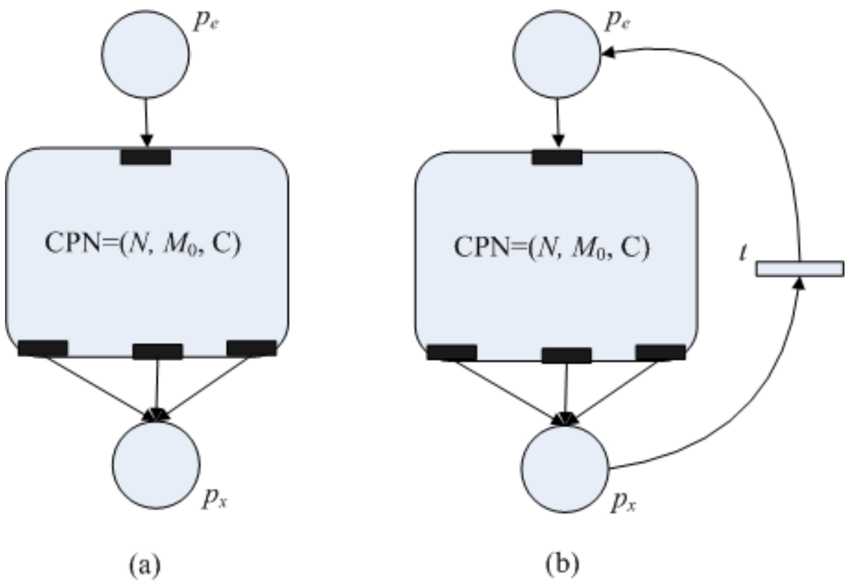


Contents

Colored Petri Net



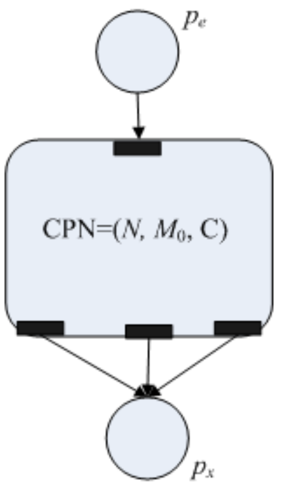
Petri Net Process



$M_e = M_0 + p_e$

$M_x = M_0 + p_x$

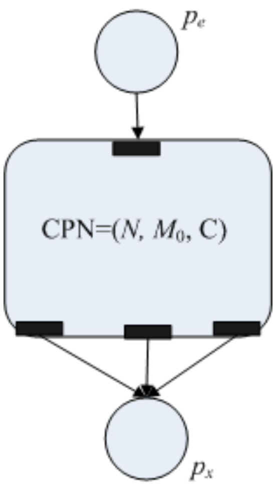
Discussion on PNP



1) Uniqueness of the entry place and exit place: by definition, a CPNP has only one entry place and one exit place. This uniqueness assumption is mainly for the consistency in modeling and convenience in creating composite processes under the various operators. For modeling real-life problems, the case of multiple entries (resp., exits) can be easily converted to the case of single entry (resp., exit) by creating a super entry (resp., exit) place and controlling the firing of its output (resp., input) places.

2) Role of the exit place p_x: in a Petri net model like CPNP, p_x is simply a sink place, indicating the location where the control flow may leave the process after one cycle of executions.

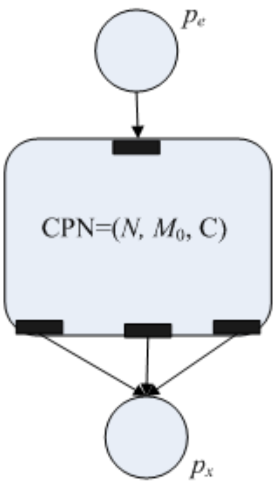
Discussion on PNP



3) M_0 represents a token distribution assigned to the set of places P before execution of process B starts. Those places having tokens serve certain 'controlling' purposes. For example, they may represent some system resources that are available before B starts its execution. The association of a static marking M_0 with B, where $M_0 \neq \emptyset$, is a special assumption of CPNP. It greatly enlarges the scope of application of CPNP.

4) Proper initiation: it is not guaranteed that (B, M_e) can always be executed. However, if B can ever start firing, it must start at M_e and not at any other marking.

Discussion on PNP



5) Deadness of Static Marking: together with the previous condition, this assumption implies that given a static marking, a process can only be initiated as follows:

- A process can start execution only after some tokens have been deposited into its entry place p_e.
- Without this deposit of tokens, (B, M_0) cannot 'self-start'. This reflects the realistic requirement that a process cannot start by itself. In order to start, it must be called by another process or by itself (i.e. recursively).

Contents

Completeness

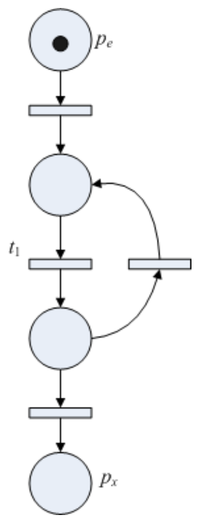
Definition 8 (Completeness). *Suppose a security policy is specified with a CPNP* $B = (CPN, p_e, p_x)$*. The policy* B *is complete if for any initial marking* M_e*, there exists a marking* $M_x = M + p_x$ *which is reachable from* M_e*.*

A security policy is decision complete (or simply complete) if it computes at least one decision for every incoming request. This property is called totality in [25] and [26].

Termination

Definition 9 (Termination). *Suppose a security policy is specified with a CPNP $B = (CPN, p_e, p_x)$. The policy B is terminating (or strongly terminating) iff B has no infinite firing sequences for any initial marking. B is said weakly terminating iff for any initial marking, B has at least one finite firing sequence. If B terminates with an exit marking $M_x = M_0 + p_x$, B is called properly terminating.*

Discussion on Termination



If B is weakly terminating, it may have infinite firing sequence(s) but must terminate in some cases

Strong termination requires that the policy always terminates with a finite number of firing steps

While properly terminating requires the policy to terminate (strongly or weakly) and to reach a special exit state.

Discussion on Proper Termination

In general, proper termination by itself does not guarantee that a process can always terminate. It just requires a process to be at the exit state $M_x = M_0 + p_x$ whenever a token has been deposited into p_x. Proper termination models the well-known `memory less' property of a software process that it should return to its initial `ready' state after having completed a cycle of execution.

Together with the Deadness of Static Marking condition, proper termination guarantees that no transition can be fired when p_x gets a token. This follows from the fact, whenever p_x gets a token, the system reaches a dead marking because M_0 in the internal CPN is dead.

The Deadness of Static Marking condition, together with the properties of Proper Initiation and proper termination, guarantee that a CPNP is non-reenterable. This means that, once having been initiated, a CPNP cannot be initiated again until its previous execution cycle has been completed. In general, to avoid mixing two independent execution cycles of a PNP, one either has to use colored Petri nets or control the procedure of entering into the process.

Proposition 1. *Suppose a security policy is specified with a CPNP $B = (CPN, p_e, p_x)$. The following properties are true:*

1. *If B is almost live, B is complete;*
2. *If B terminates properly, B is complete;*
3. *If B can terminate from any reachable marking and is deadlock-free, then B is complete;*
4. *If B is complete and almost bounded, then it is (strongly or weakly) terminating.*
5. *If B is almost live and bounded, then it is terminating.*
6. *If B is almost live and reversible, then it is properly terminating.*

Consistency

Definition 10 (Consistency). *Suppose a security policy is specified with a CPNP $B = (CPN, p_e, p_x)$. Then the policy B is consistent iff for any request $M_e = M_0 + p_e$, all reachable markings M_i satisfy that $|M_i(p_x)| \leq 1$ and for any exit markings M_j and M_k, $M_j(p_x) = M_k(p_x)$.*

Consistency implies that for any request, the policy returns at most one decision. According to the above definition, all reachable markings can have at most one token ($|M_i(p_x)| \leq 1$) with a unique identical color in place p_x (since $M_j(p_x) = M_k(p_x)$). Note that when the CPNP does not terminate, the decision place p_x will not be marked, so the consistency property is trivially satisfied.

Consistency

Proposition 2. *Suppose a security policy is specified with a CPNP B. Then, B is consistent if*

at most one of the following state equations is satisfied:
$M_i = M_e + V\mu_i$, *where* $|M_i(p_x)| = 1, i = 1, 2, \ldots$
and none of the following state equations is satisfied:
$M_i = M_e + V\mu_i$, *where* $|M_i(p_x)| > 1, i = 1, 2, \ldots$.

Confluence

Definition 11 (Confluence). *Suppose a security policy is specified with a CPNP $B = (CPN, p_e, p_x)$. The policy B is confluent iff for any initial marking $M_e = M_0 + p_e$ and any two reachable markings $M_i, M_j \in R(B, M_e)$, there exists a reachable marking M_c in B such that $M_c \in R(B, M_i) \cap R(B, M_j)$.*

A home space of B, denoted by HS, is a set of markings, such that for any $M_i, M_j \in R(B, M_e)$, there exists at least one marking M_c in HS reachable from both M_i and M_j. If a HS contains only a single element M_c, then M_c is called a home marking of B. In other words, a home marking is reachable from any marking $M \in R(B, M_e)$.

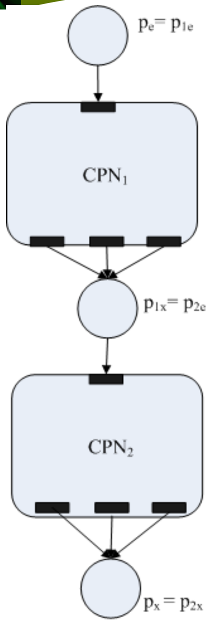
Proposition 3. *Suppose a security policy is specified with a CPNP B. Then*

1. *If a Petri net has a home marking, then it is confluent.*
2. *A safe Petri net (i.e., a PN which satisfies that the number of tokens in any place cannot exceed one for any reachable marking) has a home marking iff it is confluent.*
3. *Any confluent and strongly terminating Petri net has a unique home marking.*

Proposition 4. *Suppose a security policy is specified with a CPNP B. If B is consistent and properly terminating, then B is confluent and has a unique home marking.*

Contents

PPPA-Enable



$B_1 >> B_2$

Definition 12 (Enable (Fig. 4)). *For two processes $B_i = (P_i, T_i, F_i, W_i, M_{i0}, C_i, P_{ie}, P_{ix})(i = 1, 2)$, their composition by Enable, denoted by $B_1 \gg B_2$, is defined as the process $B = (P, T, F, W, M_0, C, p_e, p_x)$, where $P = P_1 \cup P_2$, $p_e = p_{1e}, p_x = p_{2x}$ and p_{2e} is merged with p_{1x}; $T = T_1 \cup T_2$; $F = F_1 \cup F_2$; $W = W_1 \cup W_2$; $M_0 = M_{10} \cup M_{20}$; $C = C_1 \cup C_2$.*

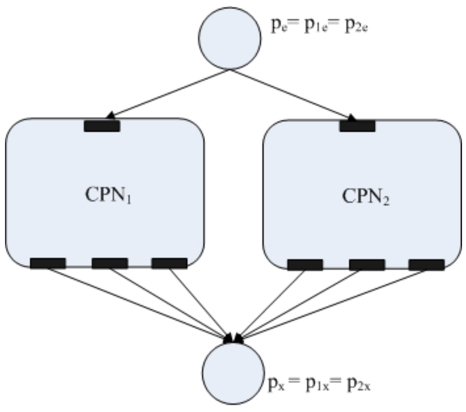
The Enable composition $B_1 \gg B_2$ models the sequential execution of two processes B_1 and B_2. That is, B_1 is first executed and B_2 is executed after the successful termination of B_1. However, if B_1 does not exit successfully, B_2 will never be activated.

Proposition for Enable

Proposition 5. *Let B be the policy obtained from two subpolicies B_1 and B_2 by applying the composition operator Enable. Then:*

1. *B is complete iff B_1 and B_2 are complete.*
2. *B is strongly (resp., properly) terminating iff B_1 and B_2 are strongly (resp., properly) terminating; B is weakly terminating iff B_1 and B_2 are weakly terminating, or B_1 is complete and B_2 is weakly terminating.*
3. *If both B_1 and B_2 are consistent, then B is consistent.*
4. *If both B_1 and B_2 are confluent, then B is confluent.*

PPPA-Choice



$B_1[]B_2$

Definition 13 (Choice (Fig. 4)). *For two processes $B_i = (P_i, T_i, F_i, W_i, M_{i0}, C_i, p_{ie}, p_{ix})$ $(i = 1, 2)$, their composition by Choice, denoted by $B_1[]B_2$, is defined as the process $B = (P, T, F, W, M_0, C, p_e, p_x)$, where $P = P_1 \cup P_2$, p_e is the place merging p_{1e} and p_{2e}, p_x is the place merging p_{1x} and p_{2x}; $T = T_1 \cup T_2$; $F = F_1 \cup F_2$; $W = W_1 \cup W_2$; $M_0 = M_{10} \cup M_{20}$; $C = C_1 \cup C_2$.*

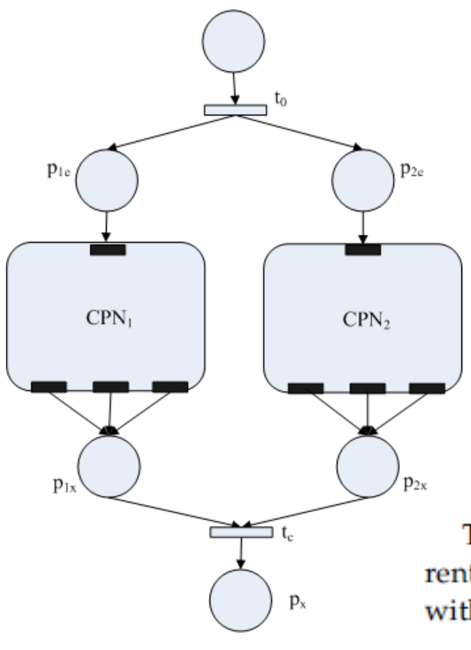
The Choice composition $B_1[]B_2$ models the arbitrary selection for execution between two processes B_1 and B_2.

Proposition for Choice

Proposition 6. *Let the policy B be obtained from two sub-policies B_1 and B_2 by applying the composition operator Choice. Then:*

1. *B is complete iff B_1 and B_2 are complete.*
2. *B is strongly (resp., weakly, properly) terminating iff B_1 and B_2 are strongly (resp., weakly, properly) terminating.*
3. *B is consistent if B_1 and B_2 are consistent and output the same colored token in the exit places.*
4. *B is not always confluent even if both B_1 and B_2 are confluent. B is confluent if B_1 and B_2 are properly terminating and output the same token in their exit place.*

PPPA-Interleave



$B_1|||B_2$

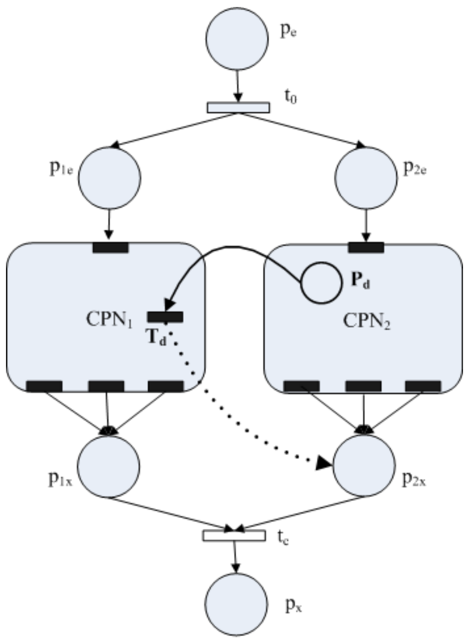
Definition 14 (Interleave (Fig. 5)). *For two processes $B_i = (P_i, T_i, F_i, W_i, M_{i0}, C_i, P_{ie}, P_{ix})$ $(i = 1, 2)$, their composition by Interleave, denoted by $B_1|||B_2$, is defined as the process $B = (P, T, F, W, M_0, C, p_e, p_x)$, where $P = P_1 \cup P_2 \cup \{p_{1e}, p_{2e}\}$, p_e, p_x are the newly added entry place and exit place, respectively; $T = T_1 \cup T_2 \cup \{t_0, t_c\}$, where t_0, t_c are newly added transitions;*

$$\begin{aligned} F = {} & F_1 \cup F_2 \cup \{(p_e, t_0), (t_0, p_{1e}), (t_0, p_{2e}), (p_{1x}, t_c), \\ & (p_{2x}, t_c), (t_c, p_x)\}; W = W_1 \cup W_2 \cup \{W(p_e, t_0), \\ & W(t_0, p_{1e}), W(t_0, p_{2e}), W(p_{1x}, t_c), W(p_{2x}, t_c), \\ & W(t_c, p_x)\}, \end{aligned}$$

where $W(t_c, p_x)$ is a 2D vector $(W(p_{1x}, t_c), W(p_{2x}, t_c))$; $M_0 = M_{10} \cup M_{20}$; $C = C_1 \cup C_2$.

The Interleave composition $B_1|||B_2$ models the concurrent but independent execution of two processes B_1 and B_2 with synchronized exit.

PPPA-Disable



B_1 [> B_2

Definition 15 (Disable (Fig. 6)). *For two processes* $B_i = (P_i, T_i, F_i, W_i, M_{i0}, C_i, P_{ie}, P_{ix})$ $(i = 1, 2)$, *their composition by Disable, denoted by* $B_1 [> B_2$, *is defined as the process* $B = (P, T, F, W, M_0, C, p_e, p_x)$, *where* $P = P_1 \cup P_2 \cup \{p_{1e}, p_{2e}\}$, p_e, p_x *are the newly added entry place and exit place, respectively;*

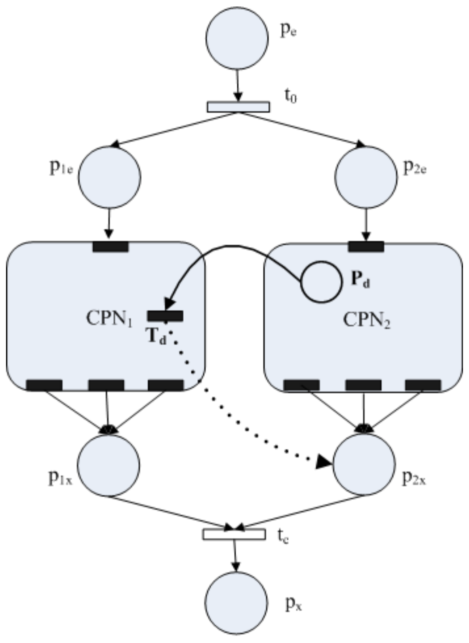
$$T = T_1 \cup T_2 \cup \{t_0, t_c\}; F = F_1 \cup F_2 \cup \{(p_e, t_0), (t_0, p_{1e}), (t_0, p_{2e}), (p_{1x}, t_c), (p_{2x}, t_c), (t_c, p_x)\} \cup \{(P_d, T_d), (T_d, p_{2x})\},$$

where

$$P_d \subseteq P_2, T_d \subseteq T_1; W = W_1 \cup W_2 \cup \{W(p_e, t_0), W(t_0, p_{1e}), W(t_0, p_{2e}), W(p_{1x}, t_c), W(p_{2x}, t_c), W(t_c, p_x)\} \cup \{W(P_d, T_d), W(T_d, p_{2x})\},$$

where $W(t_c, p_x)$ *is a 2D vector* $(W(p_{1x}, t_c), W(p_{2x}, t_c))$; $M_0 = M_{10} \cup M_{20}$; $C = C_1 \cup C_2$.

PPPA-Disable



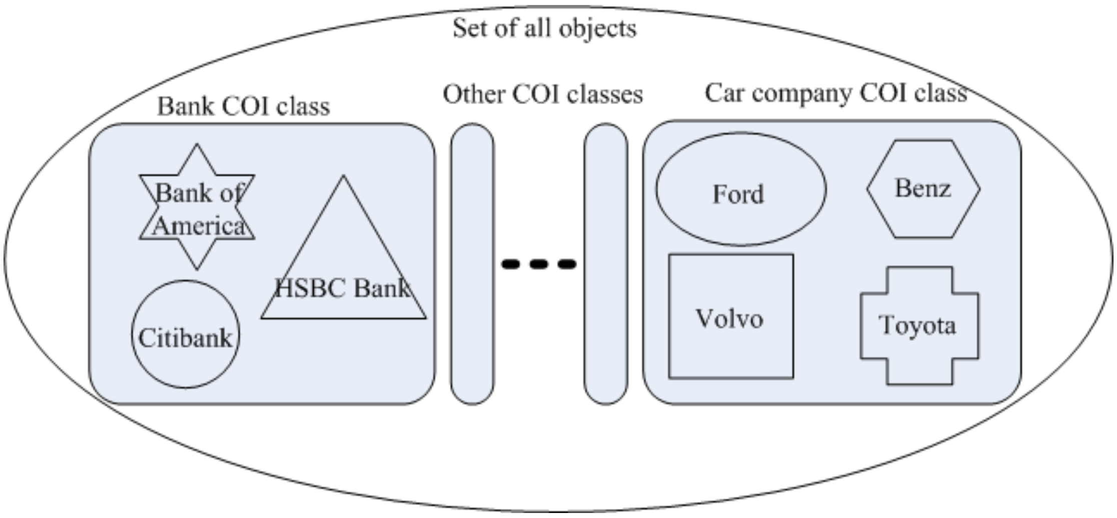
B_1 [> B_2

The Disable composition is similar to Interleave. The difference is that there exist some places P_d in B_2 which are connected to some transitions T_d in B_1 such that once T_d are fired, B_2 is dead and cannot output decisions normally. According to the policy requirement, we may add an additional arc (T_d, p_{2x}) for specifying the decisions of subpolicy B_2, if it is disabled.

Proposition 8. *Let the policy B be induced from two subpolicies B_1 and B_2 by applying the composition operator Disable. The following results are true:*

1. *Suppose the transitions in T_d are never fired in B. Then the property preservation results are the same as those for the Interleave operator in Proposition 7.*
2. *Suppose some transitions in T_d are fired and all the transitions in B_2 are disabled in B. Then:*
   - *B is complete if B_1 is complete.*
   - *B is strongly terminating if B_1 is strongly terminating; B is weakly terminating if B_1 is weakly terminating.*
   - *B is consistent if B_1 is consistent.*
   - *B is confluent if B_1 is confluent.*
3. *If B_1 and B_2 are both complete (resp., terminating), B is complete (resp., terminating), but, in general, confluence and consistency are not preserved.*
4. *Suppose B_2 satisfies the following conditions: ${}^\bullet(P_d^\bullet) = \{p_d\}$, $P_d^\bullet = {}^\bullet p_{2x}$, and $W(P_d, {}^\bullet p_{2x}) = W(P_d, T_d), W({}^\bullet p_{2x}, p_{2x}) = W(T_d, p_{2x})$. Then B is consistent (resp., confluent) iff B_1 and B_2 are consistent (resp., confluent).*

An Example for Illustration



Chinese Wall Policy

Read policy: a subject $s \in S$ can read an object $o \in O$ provided that, either there is an object $o' \in O$ such that s has accessed o' and $CD(o') = CD(o)$, or for all objects $o', o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$, where $PR(s)$ is a set of objects that s has accessed previously.

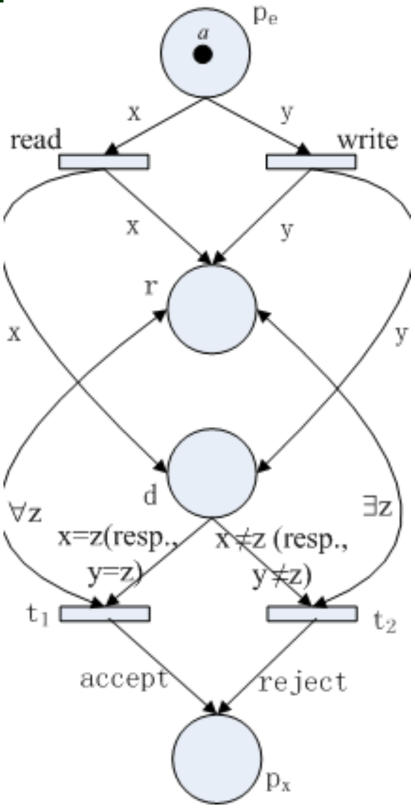
In words, a subject s is permitted to read an object o provided that, either s reads the objects all in the same CD, or reads the objects in different COIs. In the same COI, the subject cannot read objects in different CDs.

Chinese Wall Policy

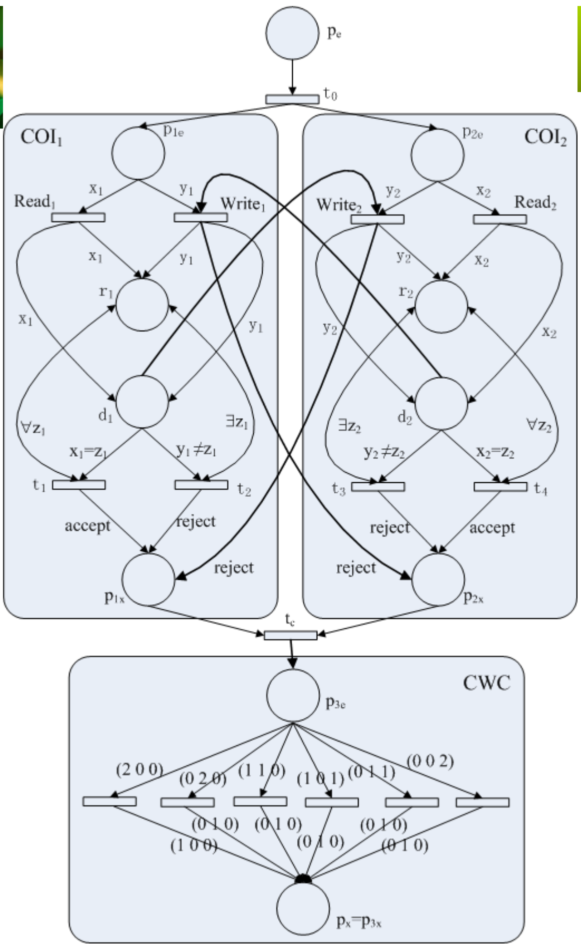
Write policy: a subject $s \in S$ may write to an object $o \in O$ provided that s is permitted to read o, and for all the objects o', s can read $o' \Rightarrow CD(o') = CD(o)$.

In words, a subject s is permitted to write an object o only when s can read o and other objects accessible by s are in the same CD with o.

PPPA –based Specification



In the Petri net model of Fig. 11, places p_e and p_x are the entry and exit interface places, respectively; place r is a record place for storing the users' previous records; place d is a place for specifying the newly requested object. Transitions *read* and *write* are operations for processing "reading" and "writing" requests, respectively. Transitions t_1 and t_2 specify "Accept" and "Reject" decisions, respectively. The detailed specification is as follows:



CWC: XACML Combiner

For a security system, in particular an access control system, the same resource may be requested by different policies and their respective decisions may be different.

XACML is a policy for conflict resolution.

It is easy to verify that both COI_1 and COI_2 are almost live, not almost bounded, deadlock free and strongly terminating. Similar to Proposition 9, CWC is complete, strongly terminating, consistent and confluent. Denoting the Chinese wall policy specification model as B = (B_1 [>B_2>> CWC, we conclude that B is complete, strongly terminating, consistent but not confluent.

Contents

Conclusion and Future Direction

It is flexible because any module (no matter whether or not it is obtained by composition of other sub-modules) can be safely replaced with an alternative design without reanalysis of the overall system architecture. Since each module is designed as a correct CPNP with a specific architecture, the replacement has the same interface and satisfies the same constraints. This feature is especially useful when we design different security policies with the same CPNP architectures

It is scalable because it allows us to analyze overall composition without the interference of internal details of the module design. Verification is done separately by checking whether each sub-module satisfies the constraints of property preservation. This significantly reduces the computing complexity and can be processed in parallel. Furthermore, our methodology is general and can be applied to a large range of security policies design, especially to a complex policy composed of some logic-related components, e.g., role-bases access control policy, task-based access control policy and rule-based policies, etc.

Conclusion and Future Direction

- Designing new combiners for restoring the unpreserved properties is another interesting future research topic.
- Specify some new specific security Properties with CPNP.
- Enhance PPPA by adding more property-preserving operators.

Thank you !