A Pragmatic Behavior Subtyping Relation Based on Both States and Actions

Wang Shengyuan (王生原), Yu Jian (喻坚) and Yuan Chongyi (袁崇义)

Received January 18, 2000, revised July 7, 2000.

Abstract. A behavior preserving relation between Petri-net systems is introduced in this paper, basing on the observability of both places and transitions, which is important in modeling the dynamic behavior of concurrent object-oriented systems with Petri nets. Each group of closely related attributes of a concurrent object is modeled by the state of a collection of observable places, and each of its methods by a group of observable transitions. The grouping distinguishes our definition from others, which makes it easy to work together with the static object models, to reuse the models and to dispel the interference among groups, thus relieving the problem of inheritance anomaly by the possibility of dividing the synchronization code into independent parts. For a formal definition of this behavior subtyping relation, Elementary Net systems, with both S-elements and T-elements labeled, are used. Then it is extended informally to state based colored Petri net systems. Finally, the background of the definitions and our future work is presented.

Keywords: Petri Net, Object Orientation, Subtyping.Relation

1. Introduction

In recent years, the integration of Petri nets with object orientation techniques has become promising [1] [2][3][4]. Parallelism, concurrency and synchronization are easy to model in terms of Petri nets, and many techniques and software tools are available for the analysis of Petri nets. These advantages have made Petri nets pretty suitable to model the dynamic behavior of objects.

However, the integration is not straightforward. One of the difficulties is that we have to face the concurrent object orientation paradigm, because Petri nets introduce concurrency into the object orientation in a natural way. The integration of concurrency and object orientation is perfectly well in anything but the inheritance. One of the main problems is the inheritance anomaly [5], in which to incrementally add code in a subclass is impossible, and some integrated code may have to be redefined, thus the benefits of inheritance are lost. The concepts about inheritance anomaly are very confusedly used in the literature [6]. To understand it clearly, it is important to distinguish between the inheritance hierarchy and the subtyping hierarchy.

In this paper, we introduce a behavior preserving relation between Petri Net systems, which is to be used as a subtyping relation between net-based objects. The main idea is (1) to observe both the external actions (T-elements) and the external states (S-elements), (2) to separate the external states into groups (divide and concur), and (3) to build a simulation relation based on the groups of states. The latter two distinguish our definition from others.

Following are the primary considerations in the definition of a subtyping relation. (1) It should have the sufficient ability of modeling the dynamic behavior of objects. (2) The pragmatic incremental inheritance relations can be easily founded. (3) The interface style of the net-based objects will be compatible with the object-orientation style. (4) It should emphasize that the S-elements and T-elements be equally important in a Petri Net system.

The usual methods to define a subtyping relation between net systems are through behavior simulation. Ref. [7] is a good survey for the simulation relations between net-based systems. To define subtyping relations from behavior simulation relations, the main decisions include (1) the observation of actions [3][4], or states [1], or both; (2) expressive or pragmatic; (3) action blocking, or action hiding, or hybrid [8]. Our

Supported by the National Natural Science Foundation of China under grant No. 69973003, and by the China NKBRSF (973) under grant G1999032706.

decisions are discussed in section 3 and section 5.

An elementary net system with both S-elements and T-elements labeled, defined in the section 2, is used to introduce the subtyping relation formally in section3. Then the subtyping notion is extended informally to the state based colored Petri net systems in section 4. And finally in section 5, the background of the notion and our future work are presented.

2. ST-Labeled Elementary Net System

2.1 Preliminaries about EN Systems

Definition 2.1.1 (*Net*) A *net* is a triple $N = (S, T; F)$, where

(1) S and T are sets such that: $S \cap T = \phi$ and $S \cup T \neq \phi$.

(2) $F \subseteq S \times T \cup T \times S$ is such that: $dom(F) \cup cod(F) = S \cup T$, where

$dom(F) = \{x \mid \exists y: (x,y) \in F\}$, and $cod(F) = \{y \mid \exists x: (x, y) \in F\}$,

S is the *set of S-elements*. T is the *set of T-elements*. And F is the *flow relation*.

For $x \in S \cup T$, ${}^\bullet x = \{y \mid (y, x) \in F\}$ is *the set of pre-elements* of x, and $x^\bullet = \{y \mid (x, y) \in F\}$ is *the set of post-elements* of x. And for $X \subseteq S \cup T$, ${}^\bullet X = \cup_{a \in X} {}^\bullet a$ and $X^\bullet = \cup_{a \in X} a^\bullet$

Definition 2.1.2 (*Elementary Net System*) A *elementary net system*, abbreviated as EN system, is a 4-tuple $N = (B, E; F, c_{in})$, where (B,E;F) is a net, and $c_{in} \subseteq B$ is called the *initial case*.

Definition 2.1.3 (*Steps*) Let $N = (B, E; F)$ be a net, $u \subseteq E$ and $u \neq \phi$, $c \subseteq B$. Then u is a *step enabled at c*, denoted by $c[u>$, iff $Ind(u) \wedge {}^\bullet u \subseteq c \wedge u^\bullet \cap c = \phi$, where

$Ind(u) \Leftrightarrow \forall e_1, e_2 \in u\ [e_1 \neq e_2 \Rightarrow ({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \phi]$.

Let $c, c' \subseteq B$. Then u is a *step leading from c to c'*, denoted by $c[u>c'$, iff $c[u>$ and $c' = (c - {}^\bullet u) \cup u^\bullet$.

Definition 2.1.4 (*Set of Cases, Set of Steps*) Let $N = (B, E; F, c_{in})$ be a EN system. The *set of cases* of N, denoted by C_N, is the smallest set such that

(1) $c_{in} \in C_N$;

(2) $\forall c \in C_N, \forall c' \subseteq B, \forall u \subseteq E (c[u>c' \Rightarrow c' \in C_N)$.

The *set of steps* of N, denoted by U_N, is the set $\{u \mid u \subseteq E \wedge \exists c \in C_N (c[u>)\}$

Let $c, c' \in C_N$, and $\omega = u_1 u_2 \cdots u_k$, where $u_1, u_2, \cdots, u_k \in U_N$. is a *step sequence leading from c to c'*, denoted by $c[\omega>c'$, iff $c[u_1>c_1, c_1[u_2>c_2, \cdots, c_{k-1}[u_k>c'$.

2.2 ST-Labeled EN Systems

Definition 2.2.1 (*ST-Labeled EN System*) A *ST-Labeled EN system*, abbreviated as STLEN system, is a tuple $\Sigma = (N, \beta)$, where

(1) $N = (B, E; F, c_{in})$ is a EN system.

(2) $\beta: B \cup E \rightarrow L \cup \{\lambda\}$ is a labeling function such that

$\forall b \in B, \forall e \in E [\beta(b) \neq \lambda \ \vee \ \beta(e) \neq \lambda \rightarrow \beta(b) \neq \beta(e)]$,

where L is the set of identifiers that range over a name space, and $\lambda \notin L$ denotes the "unobservable" S-elements or T-elements.

(3) $\lambda \notin \beta(c_{in})$.

Note that β is generally not an injection. Several observable S-elements may be labeled with a single name(identifier), and the same is true for T-elements. For $x \in B \cup E$, x is observable iff $\beta(x) \neq \lambda$.

From the definition above, we have $(\beta(B) - \{\lambda\}) \cap (\beta(E) - \{\lambda\}) = \phi$.

Let $C_\Sigma = \{c \mid c \in C_N \wedge \lambda \notin \beta(c)\}$. A case $c \in C_\Sigma$ is called an *observable case*. The initial case c_{in} is always an observable case.

For $u \in U_N$, let $\beta^+(u) = \{x \mid \exists x' \in u (x = \beta(x') \wedge x \neq \lambda)\}$. The set of *observable steps* of Σ, denoted U_Σ, is the set $\{u \mid \exists u' \in U_N (u = \beta^+(u') \wedge u \neq \phi)\}$.

It is useful to extend the function β to $\beta^*: U_N^* \rightarrow U_\Sigma^*$. For $\omega = u_1 u_2 \cdots u_k$, where $u_1, u_2, \cdots, u_k \in U_N$, $\beta^*(\omega) = \beta^+(u_1)\beta^+(u_2)\cdots\beta^+(u_k)$ in which let $\beta^+(u_i) = \varepsilon$ if $\beta^+(u_i) = \phi$, where ε be the empty string in U_N^* or U_Σ^*. Here we have $\beta^*(\varepsilon) = \varepsilon$. For $\omega \in U_N^*$, or $\omega \in U_\Sigma^*$, $\omega\varepsilon = \varepsilon\omega = \omega$. And for $\omega_1, \omega_2 \in U_N^*$, or $\omega_1, \omega_2 \in U_\Sigma^*$, $\beta^*(\omega_1\omega_2) = \beta^*(\omega_1)\beta^*(\omega_2)$.

Fig 2.1(a) is an EN system that is the behavior specification of a bounded buffer. The state of a buffer is described with three S-elements "empty", "partial" and "full" (the buffer size is assumed to be greater than 2).

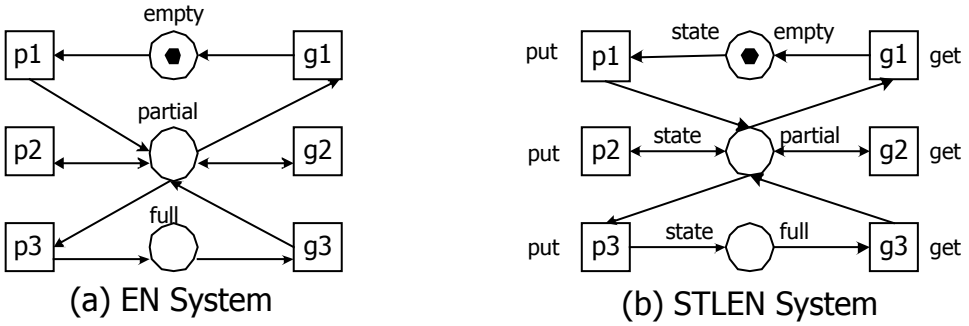

Fig 2.1 Bounded Buffer

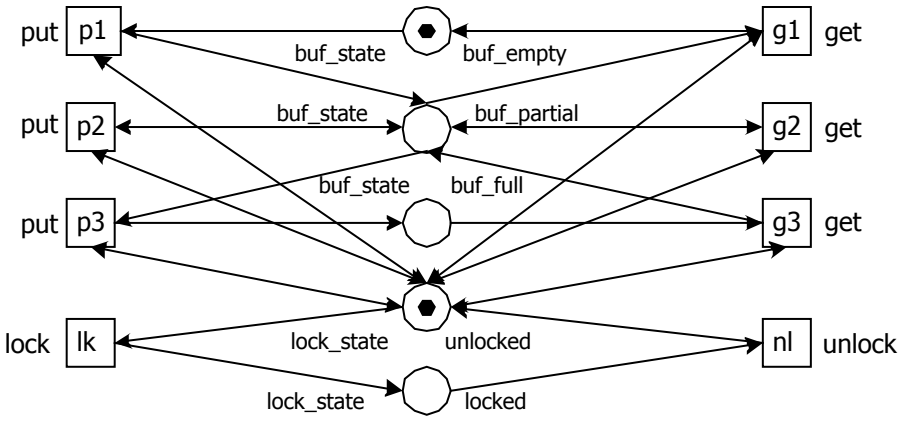

Fig 2.2 Lockable Bounded Buffer

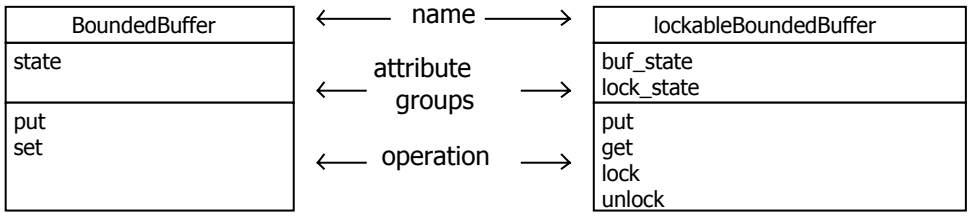

Fig 2.3 Analysis Object Models for two Classes of Buffer

The T-elements "p1", "p2" and "p3" are used to represent the possible "put" actions in different cases of the buffer state, and the T-elements "g1", "g2" and "g3" are related to "get" actions in the same meaning.

In building an object model for a bounded buffer, it is natural to use an attribute group named "state" to hold the current state of the buffer, and to let it have two public methods “put” and “get”. Then we have got an analysis object model for a bounded buffer object (or class), depicted in Fig 2.3(a).

Fig 2.1(b) is a ST-Labeled EN system whose underlying EN system is the one in fig 2.1(a). In this STLEN system, the labeling function is defined by β(“p1”) = β(“p2”) = β(“p3”) = “put”, β(“g1”) = β(“g2”) = β(“g3”) = “get”, and β(“empty”) = β(“partial”) = β(“full”) = “state”.

In diagrams of STLEN systems, the label of a T-element is drawn just outside of the box, and the label of an S-element is depicted closely on the left of the circle, and the label λ is absent.

Associate the STLEN system in fig 2.1(b) with the analysis object model in fig 2.3(a). The latter is a static object model for a bounded buffer object (or class), and the former is the dynamic object model which can serve as a specification of its dynamic behavior.

Fig 2.2 is another example of STLEN systems. A lockable bounded buffer is a bounded buffer that has

one more attribute group about the state of its "lock", and two more methods, which can change the state of this attribute. Fig 2.3(b) is the associate (static) analysis object model of a lockable bounded buffer object (or class).

2.3 Behaviors of a STLEN System

Definition 2.3.1 (*immediate step sequence*) Let $\Sigma = (N,\beta)$ be a STLEN system, and c and c' are observable. The $\omega=u_1u_2\cdots u_k$ is an *immediate step sequence leading from c to c'*, denoted by $c[\omega>>c'$ or $c[u_1u_2\cdots u_k>>c'$, iff

(1) $u_1,u_2,\cdots,u_k\in U_N$;

(2) $c[u_1>c_1, c_1[u_2>c_2, \cdots,c_{k-1}[u_k>c' \Rightarrow \forall i(1\leqslant i<k \rightarrow c_i\notin C_\Sigma)$.

Definition 2.3.2 (*externally immediate step sequence*) Let $\Sigma = (N,\beta)$ be a STLEN system, and c and c' are observable. The $\omega\in U_\Sigma^*$ is an *externally immediate step sequence leading from c to c'*, denoted by $c[(\omega>c'$, iff $\exists\omega'\in U_N^*(c[\omega'>>c'\wedge\omega=\beta^*(\omega'))$.

Most definitions about behavior properties in an EN system, such as liveliness, contact-free, deadlock-free and so on, are also useful in a STLEN system. But for the latter, some additional interesting behaviors need to be discussed. Here following are some of them.

Definition 2.3.3 (*external ST-consistency, strong external ST-consistency*) Let $\Sigma = (N,\beta)$ be a STLEN system , where $N=(B, E; F, c_{in})$ and $\beta(E)\neq\{\lambda\}$. Then

(1) Σ has *external ST-consistency* , iff $\forall c\in C_\Sigma\forall e\in E(c[e> \rightarrow \beta(e)\neq\lambda)$

(2) Σ has *strong external ST-consistency* , iff Σ has external ST-consistency , and $\forall c\in C_N\forall e\in E(\beta(e)\neq\lambda \wedge c[e> \rightarrow c\in C_\Sigma)$.

The external ST-consistency ensures that only observable actions be able to fire at each observable case, and the strong external ST-consistency demands even further that observable actions can only occur in one of the observable cases. All of the STLEN examples in this paper are strong external ST-consistency.

Definition 2.3.4 (*externally live, externally deadlock-free*) Let $\Sigma = (N,\beta)$ be a STLEN system , where $N=(B,E;F,c_{in})$ and $\beta(E)\neq\{\lambda\}$. Then

(1) Σ is *externally live* iff $\forall c\in C_\Sigma\forall e\in E(\beta(e)\neq\lambda \rightarrow\exists c'\in C_\Sigma,\omega\in U_\Sigma^*(c[(\omega>c'\wedge c'[e>))$;

(2) Σ is *externally deadlock-free* (or *weak externally ST-consistent*) iff $\forall c\in C_\Sigma\exists u\in U_N(\beta^+(u)\neq\phi\wedge c[u>)$

These two definitions are the externally extended versions of "live" and "deadlock-free" in an EN system.

Definition 2.3.5 (*drowning-free, chatting-free, divergence-free*) Let $\Sigma = (N,\beta)$ be a STLEN system. , where $N=(B,E;F,c_{in})$ and $\beta(E)\neq\{\lambda\}$. Then

(1) Σ is *drowning-free* iff $\forall c\in(C_N-C_\Sigma)\exists c'\in C_\Sigma, \omega\in U_N^*(c[\omega>c')$;

(2) Σ is *chatting-free* iff $\forall c\in C_N\exists c'\in C_N,w\in U_N^*,u\in U_N(c[w>c'\wedge\beta^+(u)\neq\phi\wedge c'[u>)$;

(3) Σ is *divergence-free* iff Σ is drowning-free and Σ is chatting-free.

The drowning-free property ensures that each internal case has always chances leading to observable ones. The chatting-free property means in any case there are always opportunities to accept external actions. But the inevitability can't be obtained except for building some fairness assumption.

3 Behavior Subtyping Relation

In this paper, the behavior subtyping means that the objects of the subtype behave the same as those of the supertype whenever the former appear in the context of the latter, which conforms to the substitutability principle [9].

3.1 ST-subtyping Relation

In the following definition, $\Sigma_1 \leq \Sigma_2$ ensures that Σ_2 behaves like Σ_1 if the external actions of Σ_2 not observable in Σ_1 are disabled. So Σ_1 can be used to specify the supertype objects, and Σ_2 to specify the subtype objects.

Besides the observable T-elements, in this paper, the observable S-elements are also labeled. So the observable S-elements are divided into groups, each of which can be associated with a set of closely related

attributes of the specified object. The state of an object is the combination of the states of their groups. To preserve the behavior of Σ_1, Σ_2 has to possess stronger state expressiveness for each of their corresponding groups, and needs to simulate the state changes of Σ_1 for each of their corresponding externally immediate step sequences. The grouping of S-elements and the group-based simulation relation make the definition different from others in the literature, which makes the modeling pragmatic by approaching the common object modeling practices in object interface styles.

Definition 3.1.1 (*ST-subtyping relation*) Let $\Sigma_1=(N_1,\beta_1)$ and $\Sigma_2=(N_2,\beta_2)$ be STLEN systems, where $N_1 = (B_1, E_1; F_1, c_{in1})$, $N_2 = (B_2, E_2; F_2, c_{in2})$, β_1: $B_1 \cup E_1 \to L_1 \cup \{\lambda\}$ and β_2: $B_2 \cup E_2 \to L_2 \cup \{\lambda\}$ are labeling functions. Then there is a ST-subtyping from Σ_1 to Σ_2, denoted by $\Sigma_1 \le \Sigma_2$, iff $\exists g{:}\beta_1(B_1)\to\beta_2(B_2)$, $\exists h{:}\beta_1(E_1)\to\beta_2(E_2)$, $\exists R \subseteq \cup_{a\in A}\rho(\beta_1^{-1}(a))\times\rho(\beta_2^{-1}(g(a)))$,where $A=\beta_1^{+}(B_1)$, such that

(1)g, h are injections with $g(\lambda)=\lambda$ and $h(\lambda)=\lambda$;

(2)$\psi(c_{in1},c_{in2})$;

(3)For $c_1\in C_{\Sigma 1}$, $c_2\in C_{\Sigma 2}$, satisfying $\psi(c_1,c_2)$, such that

$$\forall\omega_1\in U_{\Sigma 1}^{*}\,\forall c_1'\in C_{\Sigma 1}(c_1[(\omega_1>c_1'\to\exists c_2'(c_2'\in C_{\Sigma 2}\wedge c_2[(h^{*}(\omega_1)>c_2'))\wedge\psi(c_1',c_2'),$$

and (vise versa)

$$\forall\omega_2\in U_{\Sigma 2}^{*}\,\forall c_2'\in C_{\Sigma 2}(c_2[(\omega_2>c_2'\wedge\exists\omega_1\in U_{\Sigma 1}^{*}(\omega_2=h^{*}(\omega_1))\to\exists c_1'(c_1'\in C_{\Sigma 1}\wedge c_1[(h^{*-1}(\omega_2)>c_1'))\wedge\psi(c_1',c_2'),$$

where $\psi(x,y)$: $\Leftrightarrow \forall a(a\in\beta_1^{+}(B_1)\to x\cap\beta_1^{-1}(a)\ R\ y\cap\beta_2^{-1}(g(a)))$, and $h^{*}: U_{\Sigma 1}^{*}\to U_{\Sigma 2}^{*}$ with $h^{*}(\varepsilon)=\varepsilon$, and $h^{*}(u\omega) = h(u)h^{*}(\omega)$.

It's immediate to verify that the relation $\le$ is reflexive and transitive.

Proposition 3.1.1 The relation $\le$ is a preorder.

Example 3.1.1

(1) Let Σ_1 be the STLEN system in Fig 2.1(b), and Σ_2 be the one in Fig 2.2, then $\Sigma_1 \le \Sigma_2$. In fact, let g:{state}→{buf_state, lock_state} be g(state)=buf_state, and h:{put, get}→{put, get, lock, unlock} be h(put)=put, h(get)=get, and R={<{empty}, {buf_empty}>, <{partial}, {buf_partial}>, <{full}, {buf_full}>}.

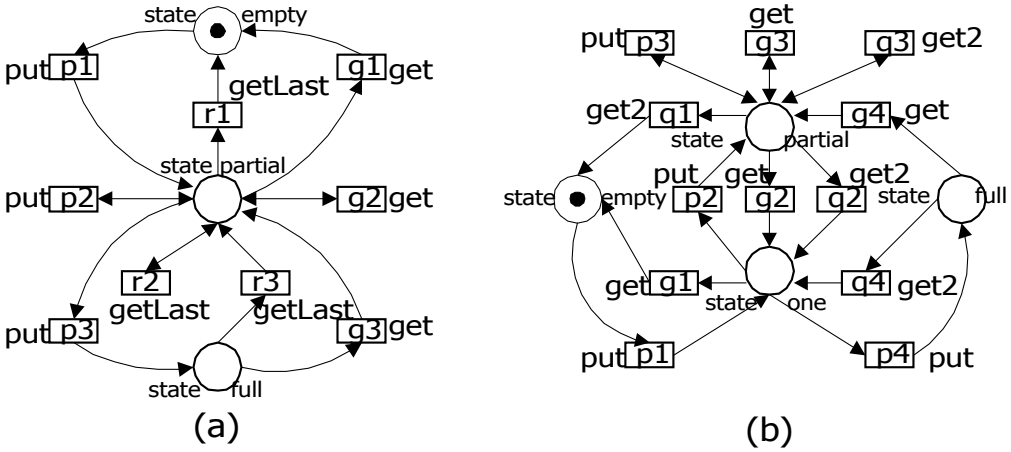

Fig 3.1 Bounded Buffers With an Additional "get" Method

(2) Let Σ_3 be the STLEN system in Fig 3.1(a) in which the operation "getLast" be added into Σ_1, then $\Sigma_1 \le \Sigma_3$. In fact, let g: {state}→{state}, and h: {put, get}→{put, get, getLast} be identity functions, and R={<{empty}, {empty}>, <{partial}, {partial}>, <{full}, {full}>}. The operation "get" removes items from the head of the buffer, and "getLast" removes items from its tail. Suppose the buffer size in Σ_1 and Σ_3 be greater than 2.

(3) Let Σ_4 be the STLEN system in Fig 3.1(b) in which the operation "get2" be added into Σ_1, then $\Sigma_1 \le \Sigma_4$. In fact, let g and h are like above, and R={<{empty}, {empty}>, <{partial}, {one}>, <{partial},

{partial}>, <{full}, {full}>}. The operation "get2" removes two items from the buffer each time. In this case, we suppose the buffer size be greater than 3. The state "partial" in Σ_1 is divided into a state "one" and a new "partial" in Σ_4, the former meaning exact one item in the buffer and the latter meaning at least 2 items in the buffer.

3.2 The Equivalence

In definition 3.1.1, we denote $\Sigma_1 \approx \Sigma_2$ (in place of $\Sigma_1 \leq \Sigma_2$) if g, h are both bijections. For relation $\approx$, the symmetricity can be verified, besides reflexivity and transitivity.

Proposition 3.2.1 The relation $\approx$ is an equivalence.

There are not many net-based equivalence notions that are based on the observation of both actions and states [7]. The interface equivalence in [10] belongs to this class. The interface equivalence, denoted by ~if~, is defined by the bisimulation between C/E systems. It's not possible to compare directly the relation $\approx$ with ~if~, because the labeling function in the latter is an injection, but not in the former. In the assumption that the underlying net system is STLEN systems for both, and the labeling function satisfies $|\beta(B)-\{\lambda\}|=1$, we have that $\Sigma_1 \sim if \sim \Sigma_2 \Rightarrow \Sigma_1 \approx \Sigma_2$.

The preserving of behavior properties of the preorder $\leq$ or the equivalence $\approx$ will not be completely discussed in this paper. But it is straightforward to prove following proposition.

Proposition 3.2.2 Let $\Sigma_1 \approx \Sigma_2$ ($\Sigma_1 \leq \Sigma_2$) be defined as above. If Σ_1 and Σ_2 are both drowning-free, then

(1) If Σ_1 is externally deadlock-free, then Σ_2 is also externally deadlock-free (in the context of Σ_1).

(2) If Σ_1 is externally live, then Σ_2 is also externally live (in the context of Σ_1).

4. Extending the Notion to State-based Colored Petri Net Systems

In a STLEN system, the S-elements, by labeling, are divided into groups. Each state of a group is decided by the states of the S-elements within the group. In this section, the STLEN system will be replaced

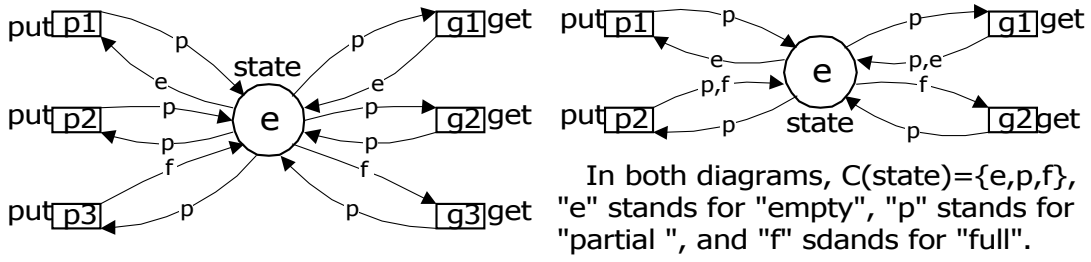

Fig 4.1 Bounded Buffer (SBLCPN Version)

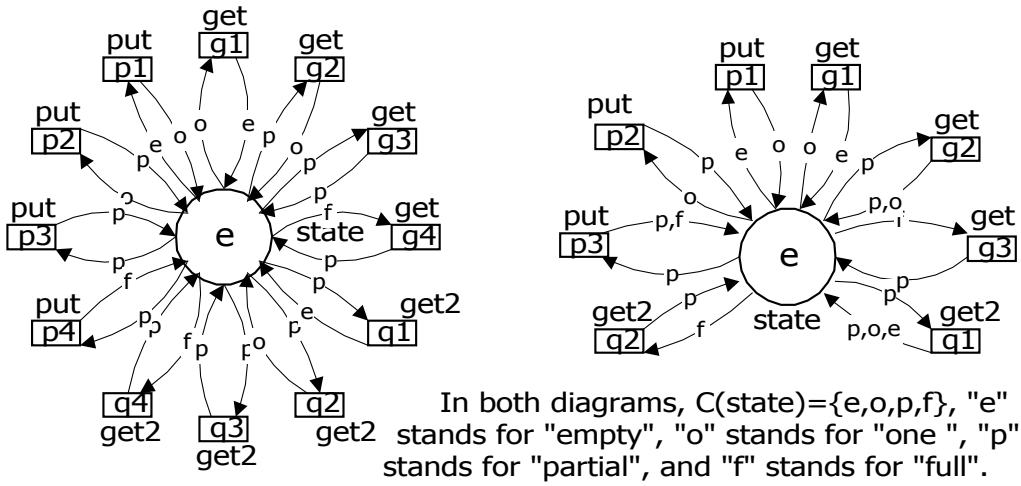

Fig 4.2 Bounded Buffer with "get2" (SBLCPN Version)

by a colored Petri Net system with the same behavior, in which the S-elements in the same group are combined into one place with each state of the group being represented by a colored token. The extension will be informally discussed in the section.

A *state based colored Petri net system* (with labeling function), abbreviated as SBLCPN, is a tuple Σ=(States, P, T, A, N, C, E, β, I), where P, T, A, N, E, I are similar with the usual Colored Petri Net (the T-guard function not used here)[11]. "States" is the set of finite sets of distinct states. C is the color function associating a place to a finite set in States. And β:P$\cup$T$\rightarrow$L$\cup\{\lambda\}$ is similar to β in STLEN systems, but with $\forall p_1,p_2 \in P(\beta(p_1)=\beta(p_2)\neq\lambda \Rightarrow p_1=p_2)$.

The Σ_1 in Fig 2.1(b) can be extended to the SBLCPN Σ_5 in the left of Fig 4.1, in which the arc expression on each arc is a singleton color name. If the arc expressions are allowed to use an "or" operator, denoted by ",", Σ_5 can be simplified to the right one in the Fig 4.1. In the same way, the Σ_4 in Fig 3.1(b) can be extended to Σ_6 in Fig 4.2. If we employ the notions in [12], the transitions with same label can also be combined into one transition, and these two net systems can be simplified further.

To extend the behavior subtyping relation in definition 3.1.1, only a little change needs to be made. For example in Σ_7 and Σ_8, let g: {state}$\rightarrow${state} and h: {put, get}$\rightarrow${put, get, get2} be identity functions, and R={<e, e>, <p, o>, <p, p>, <f, f>}. Therefore $\Sigma_5 \leq \Sigma_6$.

5. Background of the Notion and Future Work

A type is a set of instances (objects) that have some "externally observable behavior" in common [13]. Type τ is a subtype of type θ if $\tau \subseteq \theta$. Any objects of the subtype can produce the behavior common to its subtype, thus the objects of the subtype can be used in the context of objects belonging to the supertype.

The full definition of a type depends on the meaning of "externally observable behavior", and the subtyping relation between types describes how the subtype preserves the "externally observable behavior" of its supertype.

In this paper, Petri nets are used to specify the dynamic behavior of objects. The "externally observable behavior" is defined by the observability of both T-elements and S-elements. Besides the external T-elements, the external S-elements are also grouped by labeling, which distinguishes our definition from others and has at least the following benefits : (1) easy to work together with the static object models; (2) easy to reuse the models (by both inheritance and aggregation); (3) easy to dispel the interference among groups, thus relieving the problem of inheritance anomaly by the possibility of dividing the synchronization code into independent parts.

The definition of the subtyping relation $\leq$ is in terms of blocking or encapsulating actions, which means the external actions special to the subtype objects are to be inhibited in the context of the supertype objects. We have also other choices. For example, by the means of abstraction [8], the external actions special to the subtype objects will be hidden, being labeled as λ if defined in this paper, in the context of the supertype objects. Our choice is more commonly accepted in the practice.

Much work is worth paying effort to do for the notions in the paper. Some examples are the various dynamic behavior properties, the preserving of these properties, the decision problems, etc. Practically, some incremental behavior inheritance paradigms need to be developed, because it is ineffectively to use the subtyping relation directly. The related engineering methods also need to consider.

The work in this paper is a part of our recent research that is about incrementally adding the static details into the dynamic behavior models of objects. The net models are not fixed, provided that they can work together smoothly with each other. They can combine themselves with an extended UNINET model, which will be suitable in modeling both the dynamic and static aspects of objects at the detail design phase. UNINET [14] is a Petri Net version of the UNITY language [15]. The extended UNINET will possess the object orientation style.

REFERENCES

[1] E.Batiston, A,Chizzoni, Fiorella De Cindo. Inheritance and Concurrency in CLOWN. In Proceedings of

the “Application and Theory of Petri Net 1995” workshop on “object-oriented programs and models concurrency”, Torino, Italy 1995.
[2] Charles Lakos. From Colored Petri Nets to Object Petri Nets. Proceedings of 16th International Conference on the Application and Theory of Petri Nets, Lecture Notes in Computer Science 935, Turin, Italy, Springer-Verlag,1995, pp 278-297.
[3] C.Sibertin-Blanc. Cooperative Nets. Proceedings of 15th International Conference on the Application and Theory of Petri Nets. Lecture Notes in Computer Science 815, Zaragoza, Spain, Springer-Verlag, 1994, pp 471-490.
[4] D.Buchs and N.Guelfi. CO-OPN: A Concurrent Object Oriented Petri Net Approach. Proceedings of 12th International Conference on the Application and Theory of of Petri Nets, Gjern, Denmark, 1991.
[5] Satoshi Matsuoka and Akinori Yonezawa. Analysis of Inheritance Anomaly in Object-oriented Concurrent Programming Languages. In Research Directions in Concurrent Object-oriented Programming, edited by G.Agha, P.Wegner and A.Yonezawa, The MIT Press , 1993, pp.107-150.
[6] Lobel Crnogorac, Amand S.Rao, and Kotagiri Ramamohanarao. Classifying Inheritance Mechanisms In Concurrent Object-oriented Programming. LNCS 1445, ECOOP’98——Object-oriented Programming, Springer-Verlag, 1998, pp572-600.
[7] L.Pomello, G.Rozenberg and C.Simone. A Survey of Equivalence Notions for Net Base Systems, In “Advances of Petri Nets”. Lecture Notes in Computer Science 609, Springer-Verlag, 1992, pp 410-472.
[8] W.M.P. van der Aalst and T.Basten, Life-Cycle Inheritance: A Petri-Net-Based Approach, Proceedings of 18th International Conference on the Application and Theory of Petri Nets, Lecture Notes in Computer Science 1248,1997, pp 62-81.
[9] P.Wegner, S.B.Zdonik. Inheritance as an Incremental Modification Mechanism or What Like is and Isn’t Like. In ECOOP’88 Proceedings. Lecture Notes in Computer Science 322, Springer-Verlag, 1988, pp 55-77.
[10] K. Voss, Interface as a Basic Concept for Systems Specification and Verifycation, in K.Voss, H.J.Genrich, G.Rozenberg(eds.), “Concurrency and Nets”, Springer Verlag, 1987, pp.585-604.
[11] K.Jensen, Coloued Petri nets. Basic concepts, Analysis methods and Practical use, Volume 1: Concepts. EATCS monographs on Theoretical Computer Science, Springer-Verlag 1992.
[12] A.Newman, S.M.Shatz, and X.Xie. An Approach to Object System Modeling by State-Based Object Petri Nets. Journal of Circuits, Systems and Computers, Vol.8, No.1, 1998, pp.1-20.
[13] P.America. Designing an Object-oriented Programming Language with Behavioral Subtyping. In Proc. of REX School/Workshop on Foundations of Object-oriented Languages(REX/FOOL), Noordwijkerhout, the Netherlands, May, 1990, LNCS 489, Springer-verlag, 1991, pp.60-90.
[14] Yuan Chong-Yi and Qu Wan-Ling, UNITY and its Missing Structure, in Proceedings of 3rd Workshop on Advanced Parallel Processing Technologies, Changsha, China, Publishing House of Electronics Industry, 1999, pp 172-176.
[15] K.M.Chandy and J.Misra, Parallel Program Design — A Foundation, Addision-Wesley Publishing Company, 1988.