The Synthesis of Concurrent Systems Specified by Information Systems with Using the Coloured Petri Nets

Krzysztof Pancerz, Zbigniew Suraj[1]
Chair of Computer Science Foundations
University of Information Technology and Management
Sucharskiego Str. 2, 35-225 Rzeszów, Poland
e-mail: {kpancerz, zsuraj}@wenus.wsiz.rzeszow.pl

Abstract. In the paper we present a method of concurrent model construction from data based on rough set approach. The Coloured Petri Nets (*CP*-nets) have chosen as a model for concurrency.

Keywords: information system, rough sets, concurrency, Coloured Petri Nets.

1. Introduction

Information systems [4] are used for representing knowledge. In the paper we apply them as a tool for specification of concurrent systems (see also [5], [6], [7]) . We present a method for constructing from an arbitrary information system *S* its concurrent model in the form of a Coloured Petri Net CPN_S [1]. The net CPN_S exhibits the following property: the reachability set of the net CPN_S corresponds exactly to the set of all global states consistent with all rules valid in *S*. The reachability set defines an extension *S'* of *S* created by adding to *S* all new global states which are consistent with all rules true and having examples in *S*. Moreover, *S'* is the largest extension of *S* with that property. A method presented in this paper consists of two stages. In the first stage, all dependencies represented by means rules between local processes (attributes) in the system are extracted from a given set of global states (objects). In the second stage, a net corresponding to these dependencies is built.

Application of concurrent model obtained from a given information system in the form of a place-transition net (*PT*-net) has been discussed in [10], [11], [12]. On the other hand a method for constructing a concurrent model from an arbitrary information system in the form of a net with inhibitor expressions (*IE*-net) has been also discussed in [3]. However a model in the form of a *CP*-net presented in this paper is more coherent and readable than the models mentioned above.

2. Basic notions of information systems

In this section we recall some notions related to information systems.

An *information system* is a pair $S = (U, A)$, where *U* is a nonempty, finite set of *objects*, called the *universe*, *A* is a nonempty, finite set of *attributes*, i.e. $a : U \to V_a$ for $a \in A$, where V_a is called the *value set* of *a*. The set $V = \bigcup_{a \in A} V_a$ is said to be the *domain* of *A*.

Elements of *U* can be interpreted as global states of a system, however attributes can be interpreted as local processes in a given system.

Let $S = (U, A)$ be an information system. With any subset of attributes $B \subseteq A$ we associate a binary relation $ind(B)$, called an *indiscernibility relation*, which is defined by:

$$ind(B) = \{(u, u') \in U \times U : \forall_{a \in B} a(u) = a(u')\}.$$

Any information system $S = (U, A)$ determines an *information function* $Inf_A : U \to P(A \times V)$ defined by $Inf_A(u) = \{(a, a(u)) : a \in A\}$ where $V = \bigcup_{a \in A} V_a$ and *P*(*X*) denotes the powerset of *X*. The

[1] Institute of Mathematics, Rzeszów University, 35-310 Rzeszów, Rejtana Str. 16A, Poland

set $\{Inf_A(u): u \in U\}$ is denoted by INF(*S*). The values of an information function will be in the following represented by vectors of the form $(v_1,...,v_m), v_i \in V_{a_i}, i = 1,...,m$, where *m* = card(*A*). Such vectors are called *information vectors* (over *V* and *A*).

If $S = (U, A)$ than a system $S' = (U', A')$ such that $U \subseteq U'$, $A' = \{a': a \in A\}$, $a'(u) = a(u)$ for $u \in U$ and $V_a = V_{a'}$ for $a \in A$ will be referred to as a *U'*-extension of *S* (or an extension, in short). You can find more information about extensions and restrictions of information systems in [8], [9].

Any minimal subset $B \subseteq A$ such that $ind(B) = ind(A)$ is called a *reduct* in the information system *S* and denoted by *R*. The set of all reducts in *S* is denoted by $RED(S)$.

Let $S = (U, A)$ be an information system, where $A = \{a_1,...,a_m\}$, and *V* is the domain of *A*. Pairs (a,v), where $a \in A$, $v \in V$ are called *descriptors* (over *A* and *V*). By $DESC(A,V)$ we denote the set of all descriptors over *A* and *V*. Instead of (a,v) we write also $a = v$ or a_v.

A *rule* (or *positive rule*) of information system [11] is any expression of the following form:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \Rightarrow a_p = v_p,$$

where $a_p, a_{i_j} \in A$, $v_p \in V_{a_p}, v_{i_j} \in V_{a_{i_j}}$ for $j = 1,...,r$, and $\wedge$ denotes conjunction (the classical propositional operator).

The rules describe the relationship between values of the attributes in the information system. The fact that the rule is *true* in *S* we denote in the following way:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} a_p = v_p.$$

By $D(S)$ we denote the set of all rules (true) in *S*.

Let $R \subseteq D(S)$. An information vector $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_m)$ is consistent with *R* if and only if for any rule $a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} a_p = v_p$ in *R* if $\mathbf{v}_{i_j} = v_{i_j}$ for $j = 1,...,r$ then $v_p = \mathbf{v}_p$. The set of all information vectors consistent with *R* is denoted by CON(*R*).

Let $S' = (U', A')$ be a *U'*-extension of $S = (U, A)$. We say that *S'* is a *consistent extension* of *S* if and only if $D(S) \subseteq D(S')$. *S'* is a *maximal consistent extension* of *S* if and only if *S'* is a consistent extension of *S* and for any consistent extension *S"* of *S'* we have $D(S'') \subseteq D(S')$.

The set of all (optimal) rules (i.e., rules with a minimal number of descriptors on the left hand side) computed for a given reduct $R \in RED(S)$ is denoted by $OPT(S,R)$. However the set of all rules in information system *S* is denoted by $OPT(S)$. Hence

$$OPT(S) = \bigcup \{OPT(S,R): R \in RED(S)\}.$$

A method for generating the rules in this form from a given information system has been described (e.g. in [10], [11], [12]).

An *inhibitor rule* [3] of information system is any expression of the following form:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} \neg(a_p = v_p),$$

where: $a_p, a_{i_j} \in A$ and $v_p \in V_{a_p}, v_{i_j} \in V_{a_{i_j}}$ for $j = 1,...,r$, and $\neg$ denotes negation (the classical propositional operator).

The inhibitor rules describe values of a given attribute which cannot coexist when the attributes appearing on the left hand sides of these rules obtain definite values.

For each rule of the form:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} a_p = v_p$$

we can compute the set of inhibitor rules of the following form:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} \neg(a_p = v_{p_1}),$$

.....

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} \neg(a_p = v_{p_k}),$$

where $v_{p_j} \in V_{a_p} - \{v_p\}$ for $j = 1,...,k$.

Remark. It is also easy to see that each inhibitor rule of the form:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} \neg(a_p = v_p)$$

can be presented in the equivalent form in the following way:

$$a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} [(a_p = v_{p_1}) \vee ... \vee (a_p = v_{p_k})],$$

where the meaning of all symbols appearing in these both rules is such as described above.

The set of all inhibitor rules in the information system *S* corresponding to the set $OPT(S)$ is denoted by $INH(S)$.

Now we can formulate very simple and useful two properties.

Proposition 1. Let S be an information system and $OPT(S)$ be the set of all rules in system S. Moreover, let $INH(S)$ denotes the set of all inhibitor rules in the information system *S* corresponding to the set $OPT(S)$. Then CON($OPT(S)$) = CON($INH(S)$).

Proof. It follows immediately from the definitions of the considered sets of rules.

Proposition 2. Let S be an information system and *S'* its maximal consistent extension. Moreover, let $OPT(S)$ be the set of all rules true in system *S*. Then CON($OPT(S)$) = INF(*S'*).

Proof. It follows immediately from the definitions of the notions appearing in this property.

3. Basic notions of Coloured Petri Nets

In this section we recall basic notions and a notation related to Coloured Petri Nets [1].

3.1. Coloured Petri Net Structure

We use $\mathbf{N}$ to denote the set of all non-negative integers. A *multi-set m*, over a nonempty set *S*, is a function $m : S \to \mathbf{N} \cup \{0\}$. The non-negative integer $m(s) \in \mathbf{N} \cup \{0\}$ is the number of appearances of the element *s* in the multi-set *m*. The multi-set *m* can be represented by a formal sum $\sum_{s \in S} m(s)'s$.

By S_{MS} we denote the set of all multi-sets over *S*. An element $s \in S$ is said to belong to the multi-set *m* if and only if $m(s) \neq 0$. We then write $s \in m$. We use $\varnothing$ to denote the empty multi-set.

In the sequel addition, scalar multiplication, comparison, subtraction and size of multi-sets are understood in the natural way [1].

We write $Type(t) = T$ if *t* is an element of a type *T*, i.e. if $t \in T$. The element *t* can be variable, expression or token. The type of a variable *v* is denoted by $Type(v)$. We use **B** to denote the Boolean type, i.e. $\mathbf{B} = \{false, true\}$. The type of an expression *expr* is denoted by $Type(expr)$. The set of variables in an expression *expr* is denoted by $Var(expr)$. An expression without variables is said to be a *closed expression*. A *binding b* of a set of variables *V* is a function maps each variable $v \in V$ to a value $b(v) \in Type(v)$. The value obtained by evaluating an expression *expr* in a binding *b* is denoted by $expr < b >$. $Var(expr)$ is required to be a subset of the variables of *b* and the evaluation is

performed by substituting for each variable $v \in Var(expr)$ the value $b(v) \in Type(v)$ determined by the binding.

A *Coloured Petri Net* (*CP*-net) is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ satisfying the requirements below:

- Σ is a nonempty, finite set of types which are called *colour sets*,
- P is a finite set of *places*,
- T is a finite set of *transitions*,
- A is a finite set of *arcs*,
- $N : A \to (P \times T) \cup (T \times P)$ is a *node function*,
- $C : P \to \Sigma$ is a *colour function*,
- G is a *guard function*,
- E is an *arc expression function*,
- I is an *initialization function*.

The sets P, T and A exhibit the following property:

$$P \cap T = P \cap A = T \cap A = \varnothing,$$

i.e. they must be pairwise disjoint.

The multi-set in a place is called the *marking* of the place.

The guard function G is defined from T into expressions such that:

$$\underset{t \in T}{\forall} \left[Type(G(t)) = \mathbf{B} \wedge Type(Var(G(t))) \subseteq \Sigma \right].$$

It maps each transition t to an expression of Boolean type. All variables in $G(t)$ must have types that belong to Σ.

The arc expression function E is defined from A into expressions such that:

$$\underset{a \in A}{\forall} \left[Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma \right],$$

where $p(a)$ is the place of $N(a)$.

It maps each arc a into an expression which must be of type $C(p(a))_{MS}$.

The initialization function I is defined from P into closed expressions such that:

$$\underset{p \in P}{\forall} \left[Type(I(p)) = C(p)_{MS} \right].$$

It maps each place p into a closed expression which must be of type $C(p)_{MS}$, i.e. a multi-set over $C(p)$.

Let $Var(t)$ be a set of variables of transition t, such that:

$$\underset{t \in T}{\forall} Var(t) = \{ v : v \in Var(G(t)) \vee \underset{a \in A(t)}{\exists} v \in Var(E(a)) \}.$$

A *binding* of a transition t is a function b defined on $Var(t)$, such that:

1) $\underset{v \in Var(t)}{\forall} b(v) \in Type(v)$,

2) $G(t) < b >$,

where $G(t) < b >$ denotes the evaluation of the guard expression $G(t)$ in the binding b.

By $B(t)$ we denote the set of all bindings for transition t. We write bindings in the following form:

$$< v_1 = c_1, ..., v_n = c_n >,$$

where $Var(t) = \{v_1, ..., v_n\}$.

Moreover, we introduce the following notations:

$$\underset{t\in T}{\forall} A(t) = \{a \in A : N(a) \in P \times \{t\} \cup \{t\} \times P\},$$

$$\underset{(x,y)\in(P\times T)\cup(T\times P)}{\forall} E(x,y) = \sum_{a\in A(x,y)} E(a),$$

$$\underset{(x,y)\in(P\times T)\cup(T\times P)}{\forall} A(x,y) = \{a \in A : N(a) = (x,y)\}.$$

A token element is a pair (p,c) where $p \in P$ and $c \in C(p)$, while a binding element is a pair (t,b) where $t \in T$ and $b \in B(t)$. The set of all token elements is denoted by *TE*, while the set of all binding elements is denoted by *BE*.

A *marking* is a multi-set over *TE* while a *step* is nonempty and finite multi-set over *BE*.

3.2. Coloured Petri Net Dynamics

Three factors work together to determine whether a transition is enabled:

1) the multi-set of tokens in each input place of the transition,
2) the input arc inscription on each input arc connected to the transition,
3) the guard expression of the transition.

An input arc inscription is an expression on an arc that connects a place to a transition.

If each of a transition's input places contains the multi-set specified by the place's input arc inscription and the guard expression evaluates to true, the transition is enabled, and it can be fired. Otherwise it is not enabled and it cannot be fired. If the transition occurs, it removes tokens from its input places and adds tokens to its output places.

Below we present formal notions related to *CP*-net dynamics [1].

A step *Y* is *enabled* in a marking *M* if and only if the following property is satisfied:

$$\underset{p\in P}{\forall} \sum_{(t,b)\in Y} E(p,t) < b > \leq M(p).$$

When a step *Y* is enabled in a marking M_1 it may occur, changing the marking M_1 to another marking M_2, defined by:

$$\underset{p\in P}{\forall} M_2(p) = \left(M_1(p) - \sum_{(t,b)\in Y} E(p,t) < b > \right) + \sum_{(t,b)\in Y} E(t,p) < b >.$$

M_2 is *directly reachable* from M_1. We then write: $M_1[Y > M_2$.

A *finite occurrence sequence* is a sequence of markings and steps:

$$M_1[Y_1 > M_2[Y_2 > M_3 ... M_n[Y_n > M_{n+1}$$

such that $n \in \boldsymbol{N}$, and $M_i[Y_i > M_{i+1}$ for all $i \in \{1,2,...,n\}$.

A marking *M''* is *reachable* from a marking *M'* if and only if there exists a finite occurrence sequence starting in *M'* and ending in *M''*. The set of markings which are reachable from *M'* is denoted by $[M'>$. A marking is reachable if and only if it belongs to $[M_0>$. A *CP*-net is said to be *live* if, no matter what marking has been reached from the initial marking, it is possible to ultimately fire any transition of the net by progressing through some further finite occurrence sequence starting from the current marking.

Example 1. Let us consider exemplary *CP*-net shown in Figure 1. We use the CPN ML language for a description of *CP*-nets [13]. This language is used in the Design/CPN system [13] (a computer tool for Coloured Petri Nets).

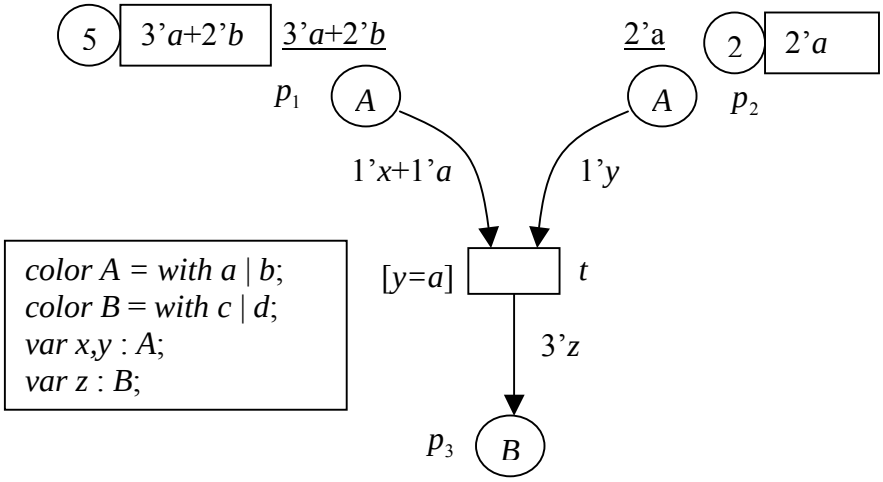


Figure 1.

We can represent the initial marking of the net in Figure 1 as a function:

$$M_0(p) = \begin{cases} 3'a + 2'b & \text{for} \quad p = p_1, \\ 2'a & \text{for} \quad p = p_2, \\ \varnothing & \text{for} \quad p = p_3. \end{cases}$$

The guard expression of the transition *t* is $[y = a]$.

Number of all bindings of the transition *t* is eight.

The bindings of the transition *t* for which the transition is enabled have the following form:

1) $< x = a, y = a, z = c >$, 2) $< x = b, y = a, z = c >$,

3) $< x = a, y = a, z = d >$, 4) $< x = b, y = a, z = d >$.

The transition *t* can be fired with each of these bindings for an accepted initial marking. We receive that the transition *t* has fired with the binding 1. The new marking of the net is shown in Figure 2.

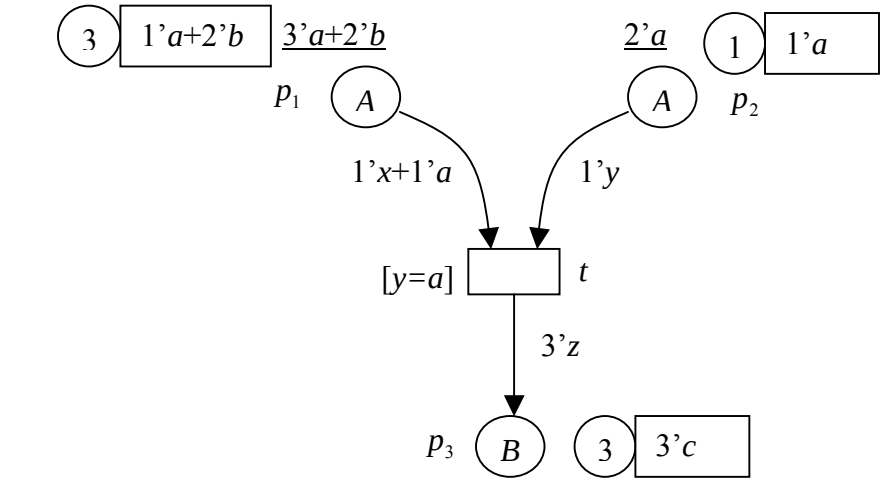


Figure 2.

4. The synthesis problem

Let $A = \{a_1,...,a_m\}$ be a nonempty, finite set of local processes (attributes). With every local process $a \in A$ is associated a finite set V_a of its internal states (values of attributes). A behavior of a modeled system can be presented in a form of a data table. Each row in the table includes record of local states of processes from *A*, and each record is labeled by an element from the set *U* of global states (objects) of the system. The columns in the table are labeled by processes. A pair (U, A) is an information system and it is denoted by *S*. The synthesis problem we can formulate as follows. Construct for a given information system *S* its concurrent model in the form of *CP*-net CPN_S with the following property: the reachability set of markings $[M_0 >$ of CPN_S defines an extension *S'* of *S* created by adding to *S* all new global states corresponding to markings from $[M_0 >$, where M_0 denotes an initial marking of the net. Moreover, *S'* is the largest extension of *S* with that property. The initial marking of CPN_S corresponds to any global state of *S*.

5. The solution of the synthesis problem

In this section we present the solution of the synthesis problem in the form of a *CP*-net constructed on the base rules extracted from a given information system.

In the case of low-level Petri nets a construction of a concurrent model by using the positive rules (see e.g. [10], [11]) is more complicated than a construction of such model on the base of the inhibitor rules [3]. However in the case of *CP*-nets it is possible to construct relatively simply a concurrent model on the base of the positive rules as well as the inhibitor rules. For *CP*-nets the both models are similar with respect to the structure of the resulting net. The difference between them concerns solely the form of logical expressions assigned to the transitions as the guard expressions. In fact, the both logical expressions are equivalent in the logical sense.

Now we are ready to describe an algorithm for constructing a concurrent model corresponding to a given information system.

Let $S = (U, A)$ be an information system.

ALGORITHM for constructing a concurrent model CPN_S of S:

`Input:` An information system *S*.

`Output:` CPN_S - the concurrent model of *S* in the form of a *CP*-net.

`Step 1.` Extract the positive rules or the inhibitor rules from an information system *S*.

`Step 2.` Construct the net representing all local processes in a given information system *S*. Each place p_a of CPN_S corresponds to a local process $a \in A$ of *S* (i.e. the number of places in the net is equal to the number of local processes in *S*). The colour sets of places in the net are labeled by means the names of attributes (local processes) of *S*. For each place the colour set of place consists of colours labeled by means the names of local states (attribute values) of a given process (attribute). There is only one transition *t* in the constructed net. The transition *t* of the net represents the global state changes (i.e. the next state relation). The initial marking M_0 of the net corresponds to a global state $u \in U$ of *S* chosen in arbitrary way.

`Step 3.` The net obtained in `Step 2` is extended by adding the guard expression to the transition *t*. The guard expression can be determined from the positive rules or the inhibitor rules computed for a given information system *S* associated with the constructed net CPN_S.

Below we present a procedure for computing a guard expression for a given set of inhibitor rules.
Let $S=(U,A)$ be an information system and let $INH(S)$ be a set of all inhibitor rules in *S* of the form:

$$a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\underset{S}{\Rightarrow}\neg(a_p=v_{p_1}),$$

.....

$$a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\underset{S}{\Rightarrow}\neg(a_p=v_{p_k}),$$

where: $a_p,a_{i_j}\in A$, $v_p\in V_{a_p}, v_{i_j}\in V_{a_{i_j}}$ for $j=1,\ldots,r$, and $v_{p_j}\in V_{a_p}-\{v_p\}$ for $j=1,\ldots,k$.

PROCEDURE for computing a guard expression:

`Input:` A set $INH(S)$ of all inhibitor rules in *S*.

`Output:` A guard expression corresponding to $INH(S)$.

`Step 1.` Rewrite each rule from $INH(S)$ using the Boolean algebra law $[a\Rightarrow\neg b]\Leftrightarrow[\neg(a\wedge b)]$ to the form of Boolean formula as follows:

$$[a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\underset{S}{\Rightarrow}\neg(a_p=v_{p_1})]\Leftrightarrow[\neg(a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\wedge a_p=v_{p_1})],$$

.....

$$[a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\underset{S}{\Rightarrow}\neg(a_p=v_{p_k})]\Leftrightarrow\neg[(a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\wedge a_p=v_{p_k})].$$

`Step 2.` Construct the conjunction of formulas obtained in `Step 1` of the form:

$$[\neg(a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\wedge a_p=v_{p_1})]\wedge\ldots\wedge\neg[(a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\wedge a_p=v_{p_k})]$$

`Step 3.` Use De Morgan laws and the Boolean algebra law of the form $a\vee a\Leftrightarrow a$ (or dual) for simplification of the formula obtained in `Step 2.` The resulting formula in the form

$$\neg[(a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\wedge a_p=v_{p_1})\vee\ldots\vee(a_{i_1}=v_{i_1}\wedge\ldots\wedge a_{i_r}=v_{i_r}\wedge a_p=v_{p_k})]$$

is the guard expression corresponding to $INH(S)$.

Example 2. Let us consider an information system $S=(U,A)$ such that $U=\{u_1,u_2,u_3,u_4\}$, $A=\{a,b\}$ and the values of the attributes are defined as in Table 1.

Table 1

U\A	*a*	*b*
u_1	0	1
u_2	1	0
u_3	0	2
u_4	2	0

By applying methods for generating the reducts and the rules described in [10], [11], [12] for the system *S* we obtain one reduct $R_1=\{a,b\}$ and the set of all rules corresponding to all nontrivial dependencies between attributes of the following form:

$$OPT(S) = \{a_1 \underset{S}{\Rightarrow} b_0, a_2 \underset{S}{\Rightarrow} b_0, b_1 \underset{S}{\Rightarrow} a_0, b_2 \underset{S}{\Rightarrow} a_0\}.$$

On the base of this set we get the following set $INH(S)$ of inhibitor rules for this system:

$$INH(S) = \{a_1 \underset{S}{\Rightarrow} \neg b_1, a_1 \underset{S}{\Rightarrow} \neg b_2, a_2 \underset{S}{\Rightarrow} \neg b_1, a_2 \underset{S}{\Rightarrow} \neg b_2, b_1 \underset{S}{\Rightarrow} \neg a_1, b_1 \underset{S}{\Rightarrow} \neg a_2, b_2 \underset{S}{\Rightarrow} \neg a_1, b_2 \underset{S}{\Rightarrow} \neg a_2\}.$$

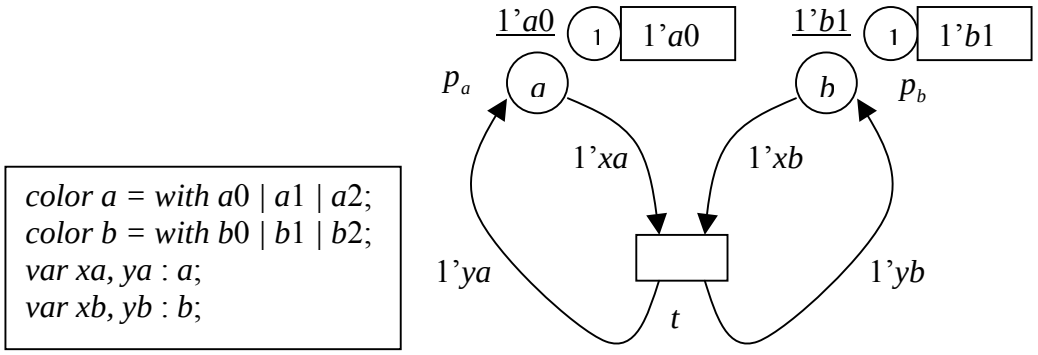
After execution of the procedure for computing a guard expression corresponding to the set $INH(S)$ we obtain the following Boolean expression:

$$\neg[(a1 \wedge b1) \vee (a1 \wedge b2) \vee (a2 \wedge b1) \vee (a2 \wedge b2)].$$

This Boolean expression is used to construct the guard expression for the sake of concurrent model construction. It has the following form:

$$[\neg((ya = a1 \wedge yb = b1) \vee (ya = a1 \wedge yb = b2) \vee (ya = a2 \wedge yb = b1) \vee (ya = a2 \wedge yb = b2))]$$

The concurrent model of *S* in the form of *CP*-net constructed by using the algorithm presented above is shown in Figure 3. The guard expression for the transition *t* (see Figure 3) has slightly different form from the presented above. It follows from formal requirements imposed by the syntax of the CPN ML language implemented into the Design/CPN system [13].



$$[not(((ya = a1)andalso(yb = b1))orelse((ya = a1)andalso(yb = b2))orelse$$
$$((ya = a2)andalso(yb = b1))orelse((ya = a2)andalso(yb = b2)))]$$

Figure 3.

The bindings of the transition *t* for which the transition is enabled have the following form:

$< xa = a0, xb = b0, ya = a0, yb = b0 >, < xa = a0, xb = b0, ya = a0, yb = b1 >,$

$< xa = a0, xb = b0, ya = a1, yb = b0 >, < xa = a0, xb = b0, ya = a0, yb = b2 >,$

$< xa = a0, xb = b0, ya = a2, yb = b0 >,$

$< xa = a0, xb = b1, ya = a0, yb = b0 >, < xa = a0, xb = b1, ya = a0, yb = b1 >,$

$< xa = a0, xb = b1, ya = a1, yb = b0 >, < xa = a0, xb = b1, ya = a0, yb = b2 >,$

$< xa = a0, xb = b1, ya = a2, yb = b0 >,$

$< xa = a0, xb = b2, ya = a0, yb = b0 >, < xa = a0, xb = b2, ya = a0, yb = b1 >,$

$< xa = a0, xb = b2, ya = a1, yb = b0 >, < xa = a0, xb = b2, ya = a0, yb = b2 >,$

$< xa = a0, xb = b2, ya = a2, yb = b0 >,$

$< xa = a1, xb = b0, ya = a0, yb = b0 >, < xa = a1, xb = b0, ya = a0, yb = b1 >,$

$< xa = a1, xb = b0, ya = a1, yb = b0 >, < xa = a1, xb = b0, ya = a0, yb = b2 >,$

$< xa = a1, xb = b0, ya = a2, yb = b0 >,$

$< xa = a1, xb = b1, ya = a0, yb = b0 >, < xa = a1, xb = b1, ya = a0, yb = b1 >,$
$< xa = a1, xb = b1, ya = a1, yb = b0 >, < xa = a1, xb = b1, ya = a0, yb = b2 >,$
$< xa = a1, xb = b1, ya = a2, yb = b0 >,$
$< xa = a1, xb = b2, ya = a0, yb = b0 >, < xa = a1, xb = b2, ya = a0, yb = b1 >,$
$< xa = a1, xb = b2, ya = a1, yb = b0 >, < xa = a1, xb = b2, ya = a0, yb = b2 >,$
$< xa = a1, xb = b2, ya = a2, yb = b0 >,$
$< xa = a2, xb = b0, ya = a0, yb = b0 >, < xa = a2, xb = b0, ya = a0, yb = b1 >,$
$< xa = a2, xb = b0, ya = a1, yb = b0 >, < xa = a2, xb = b0, ya = a0, yb = b2 >,$
$< xa = a2, xb = b0, ya = a2, yb = b0 >,$
$< xa = a2, xb = b1, ya = a0, yb = b0 >, < xa = a2, xb = b1, ya = a0, yb = b1 >,$
$< xa = a2, xb = b1, ya = a1, yb = b0 >, < xa = a2, xb = b1, ya = a0, yb = b2 >,$
$< xa = a2, xb = b1, ya = a2, yb = b0 >,$
$< xa = a2, xb = b2, ya = a0, yb = b0 >, < xa = a2, xb = b2, ya = a0, yb = b1 >,$
$< xa = a2, xb = b2, ya = a1, yb = b0 >, < xa = a2, xb = b2, ya = a0, yb = b2 >,$
$< xa = a2, xb = b2, ya = a2, yb = b0 > .$

The following markings of the constructed *CP*-net can be generated with this bindings:

Table 2

Marking \ Place	p_1	p_2
M_1	1'a0	1'b1
M_2	1'a1	1'b0
M_3	1'a0	1'b2
M_4	1'a2	1'b0
M_5	1'a0	1'b0

The set of markings of the constructed *CP*-net corresponds to all global states consistent with all rules valid in information system *S*. Hence the largest extension of *S* has the following form:

Table 3

*U**A*	*a*	*b*
u_1	0	1
u_2	1	0
u_3	0	2
u_4	2	0
u_5	0	0

It is worth to observe that the object u_5 corresponding to the marking M_5 of the constructed *CP*-net is new. Moreover, the information vector determined by the object u_5 is consistent with $INH(S)$.

6. The solution properties of the synthesis problem

This section presents a relationship between the reachability set $[M_0>$ of the *CP*-net CPN_S constructed for a given information system *S* and the set of all information vectors consistent with rules true in *S*.

The algorithm described in Section 5 for constructing from an arbitrary information system *S* its concurrent model in the form of a *CP*-net has the property which we express formally in the form of the following theorems.

Theorem 1. Let *S* be an information system and *S'* its maximal consistent extension. Moreover, let CPN_S be a *CP*-net constructed by using our algorithm presented in Section 5. Then the reachability set $[M_0>$ of CPN_S is equal to the set of all information vectors INF(*S'*), i.e., $[M_0>$ = INF(*S'*).

Proof. It follows from Property 1 and Property 2, and the fact that in our construction of *CP*-net we eliminate all markings defining information vectors which are inconsistent with CON($INH(S)$) and only those markings.

Theorem 2. Let *S* be an information system and CPN_S its concurrent model constructed by using our algorithm presented in Section 5. Then the net CPN_S is live at the initial marking M_0.

Proof. Follows from the construction of the net CPN_S.

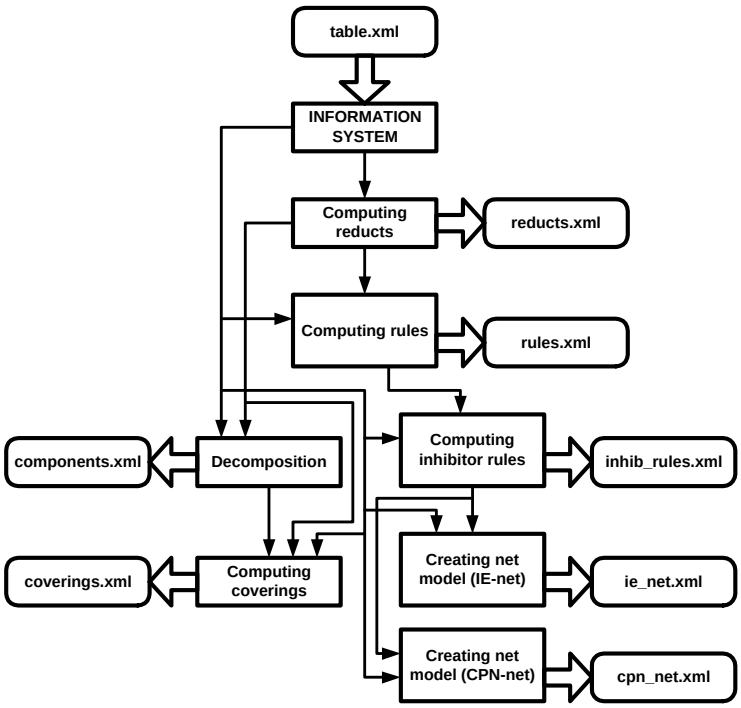


Figure 4.

7. Conclusions

The paper concerns an approach to concurrency, based on the Coloured Petri Nets and rough set theory.

CP-nets extracted from a given information system have been chosen as a model for concurrency. *CP*-nets make possible constructing more coherent and simpler models than models described in [3], [10], [11]. In the further investigations we shall study application of the presented model for synthesis of concurrent system specified by the dynamic information systems [12].

The method proposed in the paper has been implemented according to a new programming technology in the ROSECON system [2] running on IBM PC computers under Windows operating system. The ROSECON system is being developed in the Chair of Computer Science Foundations at the University of Information Technology and Management in Rzeszów. At present the ROSECON system makes possible computing reducts, rules, inhibitor rules, components and coverings of information systems specified by information tables, and creating their concurrent models in the form nets with inhibitor expressions [3] or *CP*-nets. A block diagram of the ROSECON system is shown in Figure 4.

Input and output data of the ROSECON system are written in XML format. This system is permanently evolved.

Acknowledgement. We are grateful to Professor A. Skowron for interesting suggestions about this work. The research of Zbigniew Suraj has been partially supported by grant 8 T11C 025 19 from the State Committee for Scientific Research (KBN) in Poland.

References

[1] Jensen, K.: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 1.* Springer-Verlag, Berlin Heidelberg 1992.

[2] Pancerz, K., Suraj, Z.: ROSECON – a system for automatic discovering of concurrent models from data tables. In: *Proceedings of the IX Environmental Conference on Mathematics and Informatics*, Korytnica, June 12-16, 2002, Poland (in Polish), p. 34.

[3] Pancerz, K., Suraj, Z.: From Data to Nets with Inhibitor Expressions: A Rough Set Approach. In: Z. Suraj (ed.), *Proceedings of the Sixth International Conference on Soft Computing and Distributed Processing*, Rzeszow, June 24-25, 2002, Poland, pp. 102-106.

[4] Pawlak, Z.: *Rough Sets – Theoretical Aspects of Reasoning About Data.* Kluwer Academic Publishers, Dordrecht 1991.

[5] Pawlak, Z.: Concurrent versus sequential the rough sets perspective. *Bulletin of the EATCS,* 48, 1992, pp.178-190.

[6] Pawlak, Z.: Some Remarks on Explanation of Data and Specification of Processes Concurrent. *Bulletin of International Rough Set Society*, 1-1, 1997, pp. 1-4.

[7] Peters, J.F., Skowron, A., Suraj, Z., Pedrycz, W., Ramanna, S.: Approximate Real-Time Decision Making: Concepts and Rough Fuzzy Petri Net Models. *International Journal of Intelligent Systems*, 14-4, 1998, pp. 4-37.

[8] Rząsa, W., Suraj, Z.: A new method for determining of extension and restriction of information system. In: *Proceedings of the Third International Conference a Rough Sets and Current Trends in Computing*, Penn State Great Valley, October 14-16, 2002, PA, USA.

[9] Suraj, Z.: Some Remarks on Extensions and Restrictions of Information Systems. In: W. Ziarko, Y. Yao (eds.), *Rough Sets and Current Trends in Computing, Lecture Notes in Artificial Intelligence*, Vol. 2005, Springer, Berlin 2001, pp. 204-211.

[10] Suraj, Z.: Discovery of Concurrent Data Models from Experimental Tables: A Rough Set Approach. *Fundamenta Informaticae*, Vol. 28, No. 3, 4, December 1996, pp. 353 – 376.

[11] Suraj, Z.: Rough Set Methods for the Synthesis and Analysis of Concurrent Processes. In: L. Polkowski, S. Tsumoto, T.Y. Lin (eds.) *Rough Set Methods and Applications*, Springer, Berlin 2000, pp. 379-488.

[12] Suraj, Z.: The Synthesis Problem of Concurrent Systems Specified by Dynamic Information Systems. In: L. Polkowski and A. Skowron (eds.), *Rough Sets in Knowledge Discovery*, 2, Physica-Verlag, Berlin, 1998, pp. 418-448.

[13] http://www.daimi.au.dk/designCPN/