• 1400008538
Còpia 1

Fimite memory devices in CSP

Joaquim Gabarró
María José Serna

Report LSI–90–40

8203

Finite Memory Devices in CSP

J. Gabarró * M.J. Serna*

Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo 5 , 08028 Barcelona
Spain

Abstract: It is often said that a state based approach to CSP is inadequate, however we present here some (theoretical) hints against this assertion. A new class of processes modelled by finite memory devices are considered. These devices (called here CSP automata) allow both: deal with the different kinds of nondeterminism at a state level and model misbehaviours due to divergences. They are well adapted to the semantics of failures plus divergences. As CSP is independent of branching time CSP-automata can be determinized. Furthermore we show that an extension of the classical automata's morphism is equivalent to refinement between processes. That allow us to define canonical forms through minimization. These processes can also be characterized by a set of recursive equations so called linear systems. These processes are stable under nondeterminism, change of symbol, prefixing and interleaving.

Keywords: CSP, failures model, normal forms, state-based-approach, finite automata, minimization, refinement, morphism, linear systems.

* Research supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

1. Introduction

It is well known that CSP semantics is based on traces (or failures) and often it is said that a state based approach seems to be inadequate. We give here some hints against this assertion. We present a finite memory device so called *finite CSP automata* matching adequately with the semantics of failures and divergence. These automata have remarkable properties like determinization and minimization. The techniques developped are very close to those of finite automata. This is a good phenomenon that permits to obtain short and intuitive proof of some facts. Let us present in more detail our work.

To build a bridge between failures semantics and finite state automata we take some shortcuts on notations describing processes. Let us explain them over an example. Let $P = (A, F, D)$ be a process [Hoa85] such that:

- The events are the set $A = \{a, b, c\}$.
- The failures F are given by the union of the following sets:

$$\{s, X \mid s \in a^* \text{ and } X \in \mathcal{P}(\{b, c\})\}$$
$$\{s, X \mid s \in a^*b^+ \text{ and } X \in \mathcal{P}(\{a, b\}) \cup \mathcal{P}(\{a, c\})\}$$
$$\{s, X \mid s \in a^+A^* \text{ and } X \in \mathcal{P}(\{a, b, c\})\}$$
$$\{s, X \mid s \in a^*b^+cA^* \text{ and } X \in \mathcal{P}(\{a, b, c\})\}$$

- The divergences are $D = a^+A^* + a^*b^+cA^*$.

First of all we adopt a more condensed notation describing failures. For example the set $\{s, X \mid s \in a^* \text{ and } X \in \mathcal{P}(\{b, c\})\}$ is represented without ambiguity as $a^*\underline{bc}$, with this notation:

$$F = a^*\underline{bc} + a^*b^+(\underline{ac} + \underline{ab}) + a^+A^*\underline{A} + a^*b^+cA^*\underline{A}$$

Second, we can "mark" explicitely the divergences with an $\uparrow$ as:

$$D = a^+A^* \uparrow + a^*b^+cA^* \uparrow$$

Finally the whole process P can be noted without any ambiguity as:

$$P = a^*\underline{bc} + a^*b^+(\underline{ac} + \underline{ab}) + a^+A^*\underline{A} + a^*b^+cA^*\underline{A} + a^+A^* \uparrow + a^*b^+cA^* \uparrow$$

As $Chaos = A^*\underline{A} + A^*\uparrow$ the process can be rewritten as:

$$P = a^*\underline{bc} + a^*b^+(\underline{ac} + \underline{ab}) + (a^+ + a^*b^+c)Chaos$$

As we are interested in a state based approach to CSP, let us recall some definitions of automata theory [Eil74]. A *nondeterministic automaton* is a tuple $M =< A, Q, T, Q_0 >$. The set A is a finite *set of events*, Q is the *set of states*, $T \subseteq Q \times A \times Q$ is the *set of*

transitions and $Q_0 \subseteq Q$ are the *initial states*. As every procees contains $<>$ as a trace we assume $Q_0 \neq \{\}$. Given two states p and q we note:

$$L(p,q) = \{w \mid (p,w,q) \text{ is a path from } p \text{ to } q\}$$

In the case of deterministic automaton the set of transitions become a partial function (we note (p,w,q) as $p \cdot w = q$) and there is only one initial state as q_0.

There are works modelling CSP processes by state devices [Jos88, HeJ89]. In order to explain carefully our model, we consider briefly the approach taken by Josephs [Jo88] (dealing only with divergence free processes). A process is modelled by a nondeterministic automaton $M =< A, Q, T, Q_0 >$. Fixed an state and an environment, the system blocks iff it cannot evolve in the direction given by the environment. Formally we attach to any state q a unique maximal set of refusals

$$R(q) = \overline{next}(q) = \{x \mid \neg\exists q' : (q, xq') \in T\}$$

and with the previous notations the process defined by M is:

$$failures(M) = \bigcup_{(q,p)\in Q_0 \times Q} L(q,p)\underline{R(q)}$$

This model has two limitations. Firstly, it seems difficult to have a clear treatement of the different kinds of nondeterminism at a state level. Secondly, we cannot deal with misbehaviours due to divergences.

To improve these limitations we define a new class of finite memory device called CSP-automata. The states of these automata have additional information about possible deadlocks and misbehaviours due to divergences. Formally a CSP-automata is a triple $P =< M, R, D >$ where M is an usual nondeterministic finite automata and:

- The treatement of possible deadlocks match closely the states of M through a *refusals function* R from Q into $\mathcal{PP}(\underline{A})$. The function R associates to every state q a set of refusals $R(q) = \{R_1, R_2, \ldots\}$, this function permits a careful and economic treatement of the two different classes of non determinism at a state level.
- The misbehaviours due to divergences are treated explicitely. The automaton M can have some pathological states $D \subseteq Q$ called divergent states. These states are very strong an when the process come in never can get out. The possibility of recovery is impossible. Of course the non divergent process satisfies $D = \{\}$.

Adding this additional structure to the states, specified by R and D, we obtain automata well adapted to model phenomena of deadlock, nondeterminism and divergence. Note that in general the set Q does not need to be finite. Processes that can be modelled (as above) using a finite state automaton will be called *regular*. Thus we identify regular processes with finite state nondeterministic CSP-automata.

As the failures semantics is independent of branching time regular processes can be determinized. Furthermore we show that an extension of the classical automata's morphism is equivalent to refinement between processes. That is, given two regular process P and P' we have that $P \sqsubseteq P'$ if and only if there is a CSP-morphism from P to P'. Then the notion of refinement is equivalent to a classical notion of automata theory. It is well known that refimement are used to obtain canonical forms [Jos88]. In our case these ideas can be implemented through minimization of CSP-automata. Of course minimal CSP-automata can be of considerable size, but at least they exists.

It is possible to extend the classical theory of linear systems [Eil74] to describe processes. Using this approach we get a compact process description. Furthermore we can apply the Arden's lemma to solve recursive equations and we obtain a second approach to define processes. As these processes can be described by linear equations we call them *rational processes*. We prove that both classes are equal, then we can use a CSP-automata or a linear system to describe a regular (or rational) process. Furthermore, the class of regular processes enjoys of interesting closure properties. It is stable under nondeterminism, change of symbol, prefixing and interleaving.

2. Regular processes

We are interested in defining a finite memory device well adapted to describe CSP processes. A process P will be modelled by a *CSP-automaton*. Formally a process P is a triple $P =< M, R, D >$ where a non deterministic automaton $M =< A, Q, T, Q_0 >$, called the *base automaton*, models the finite control structure. The set of events is A, the states are the set Q, the transition relation is $T \subseteq S \times A \times S$ and the nonempty set of initial states is Q_0. The traces can be described as:

$$traces(P) = \bigcup_{(q,p) \in Q_0 \times Q} L(q,p)$$

To extend the trace model we consider the *refusals mapping* $R : Q \rightarrow \mathcal{PP}(\underline{A})$. In this mapping $\underline{A}$ is a marked copy of A. For each state $q \in Q$ the refusal set of q is denoted as $R(q) = \{R, R', \ldots\}$. This function verifies

- $\{\} \in R(q)$.
- If $R \in R(q)$ and $R' \subseteq R$ then $R' \in R(q)$.
- Given $a \in A$ and $R \in R(q)$ then $R \cup \{a\} \in R(q)$ or exists q' such that $(q, a, q') \in T$.

With the refusals mapping it is easy to prove that the following set is really a set of failures.

$$failures(P) = \bigcup_{(q,p) \in Q_0 \times Q} L(q,p)\underline{R(p)}$$

To consider divergences we mark some states $D \subseteq Q$ as *divergent* ones. Every divergent state q verifies

- For every $a \in A$, (q, a, q) belongs to T and these are the only transitions out of q.
- $R(q) = \mathcal{P}(\underline{A})$

Then the divergences set of P can be written as

$$\mathit{divergences}(P) = \bigcup_{(q,p) \in Q_0 \times D} L(q,p)$$

When the alphabet of events A is known the process $P = (A, \mathit{failures}(P), \mathit{divergences}(P))$ can be described shortly as:

$$P = \bigcup_{(q,p) \in Q_0 \times Q} L(q,p)\underline{R(p)} + \bigcup_{(q,p) \in Q_0 \times D} L(q,p)\,\mathit{Chaos}$$

Note that in general the set Q does not need to be finite. We say that a CSP process P is *regular* if P can be modelled (as above) using a finite state automaton M. Thus regular processes are identified with finite state nondeterministic CSP-automata. Let us give some examples of regular CSP processes

Examples 1: In figure 1 we show some CSP-automata. All the processes have $A = \{a, b\}$. Inital and divergent states are marked with a nonlabelled arrow and with a nonlabelled dotted arrow respectively.

- The first two automata corresponds to process *Chaos* and *Stop*. The automaton P_1 models the process $P_1 = \underline{b} + a\ \mathit{Chaos}$ defined by the equation

$$X_1 = a \rightarrow \mathit{Chaos}$$

- The automata P_2, P_3 and P_4 model the different kind of nondeterminism. P_2 corresponds to the process $P_2 = (a + b)^*\underline{\{\ \}}$ and can be defined by

$$X_2 = a \rightarrow X_2 \,\Box\, b \rightarrow X_2$$

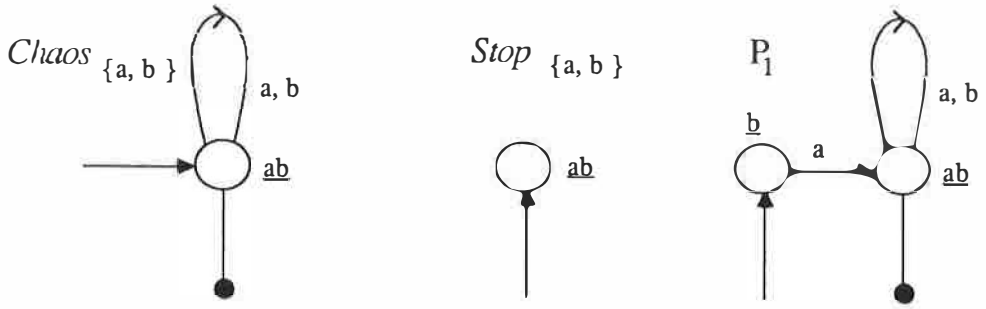
  P_3 corresponds to $P_3 = (a + b)^*(\underline{a} + \underline{b})$ and is the solution of

$$X_3 = a \rightarrow X_3 \sqcap b \rightarrow X_3$$

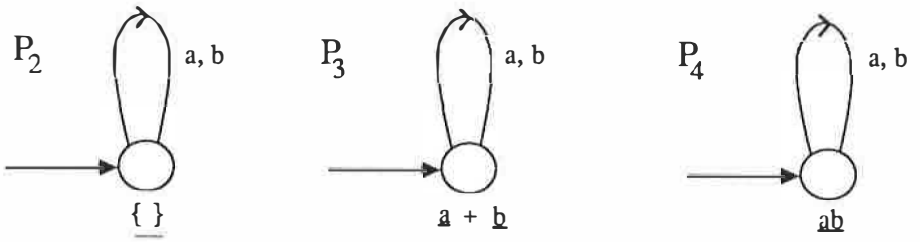
  Finally P_4 corresponds to $P_4 = (a + b)^*\underline{ab}$. It is the solution of

$$X_4 = a \rightarrow X_4 \sqcap b \rightarrow X_4 \sqcap \mathit{Stop}$$

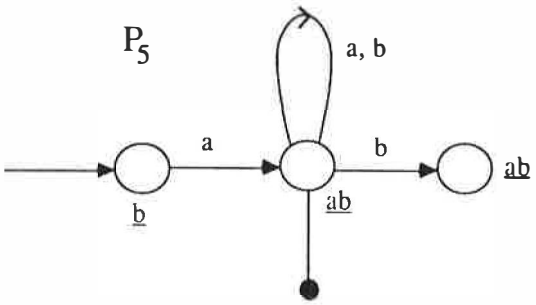
- We could try to represent $X_5 = a \rightarrow (b \rightarrow \mathit{Stop} \sqcap \mathit{Chaos})$ as the device P_5. However P_5 is not a CSP automaton. Just note that there is an arc going from a divergent state to a nondivergent state. This kind of transitions are not allowed in our formalism.



Some elementary CSP-automata



Different kinds of nondeterminism



A non CSP automaton

Figura 1

A CSP regular process will be called *deterministic* if the base automaton is deterministic. It is important to remark that deterministic regular CSP processes are not forcely deterministic process in the sense of [Hoa85]. Furthermore, as CSP process are independent of branching time, we can extend the general determinisation algorithm [Eil74] to CSP-automata as follows. Let P be the CSP-automaton $< M, R, D >$ with $M =< A, Q, T, Q_0 >$. The states of $det(P)$ are elements of $\mathcal{P}(Q)$ denoted as $Q_0, Q_1, \ldots$ The initial state is Q_0. An state Q_i

is divergent if $Q_i \cap D \neq \{\}$ otherwise is non-divergent. Given a divergent state Q_i the only transitions out of Q_i are (Q_i, a, Q_i) for every event a. For a non divergent state Q_i we have the transition (Q_i, a, Q_j) iff it exists $q \in Q_i$ and $q' \in Q_j$ such that $(q_i, a, q_j) \in T$. The refusals mapping is defined as $R(Q_i) = \cup_{q \in Q_i} R(q)$. Then we have the following result.

Theorem 2: For every CSP regular process, P, there is a CSP deterministic process, $det(P)$, such that $P = det(P)$.

Thus the class of processes that can be modelled using nondeterministic CSP automata equals the class of processes defined by deterministic CSP automata.

3. Refinement and canonical forms

We record the refinement of a process P into a process P' by writing $P \sqsubseteq P'$ (P is less defined than P'). The first condition that must be met is that both processes have the same alphabet. The second one can be stated informally as: every behaviour that is possible for P must be also possible for P'. In order to deal in a formal way with refinement we introduce the notion of CSP morphism. This notion extend the classical idea of automata morphism [Eil74]. As in classical automata theory, CSP morphisms together with minimization techniques give us minimal process that are canonical forms.

Through the section, we shall assume that regular processes are given as deterministic CSP automata. Thus we assume that the base automaton is deterministic, it has an unique initial state and the transition relation is given by a transition function.

Let us introduce the notion of CSP morphism. Let assume that $P =< M, R, D >$ with $M =< A, Q, *, q_0 >$ and $P' =< M', R', D' >$ with $M' =< A, Q', \circ, q_0' >$ are two regular processes. A partial mapping $\phi : Q \rightarrow Q'$ is called a *CSP morphism* and is denoted by $\phi : P \rightarrow P'$ if the following holds:

1. $\phi(q_0) = q_0'$.
2. If $\phi(q) \circ a$ is defined then $q * a$ is defined and $\phi(q) \circ a = \phi(q * a)$
3. For each $q \in Q$, if R' is a refusal set for $\phi(q)$ then R' is a refusal set for q. Formally, $R'(\phi(q)) \subseteq R(q)$.
4. If $\phi(q)$ is a divergent state in M' then q is a divergent state in M. Formally $\phi^{-1}(D') \subseteq D$.

As a direct consequence of the definition we have:

Lemma 3: CSP morphism composse.

Let us now characterize refinement for the class of CSP regular processes through CSP morphism.

Lemma 4: Given two regular CSP processes P, and P'. If there is a CSP morphism from P into P' then $P \sqsubseteq P'$.

Proof. First note that as the base automaton is deterministic, we can extend condition 2 of the definition of CSP morphism to words, that is: Given $s \in A^*$ if $\phi(q) \circ s$ is defined then $q * s$ is defined and $\phi(q * s) = \phi(q) \circ s$. Thus when $\underline{sR}$ is a failure of P' we have that $q'_0 \circ s$ is defined. That means that $q_0 * s$ is defined and $\phi(q_0 * s) = \phi(q_0) \circ s = q'_0 \circ s$. Then by condition 3 of the definition of CSP morphism we have that $\underline{sR}$ is a failure of P.
The same argument applies to divergences using condition 4 of the definition of CSP morphism. Thus we conclude $P \sqsubseteq P'$.□

Lemma 5: Given two regular CSP processes P, and P'. Whenever $P \sqsubseteq P'$ there is a CSP morphism from P into P'.

Proof. We define a mapping ϕ inductively as $\phi(q_0) = q'_0$, and for every state $q \in Q$ there is a $s \in A^*$ such that $q_0 * s = q$. As $P \sqsubseteq P'$ s is a trace of P', thus $q'_0 \circ s$ is defined, we define

$$\phi(q) = \phi(q_0 * s) = \phi(q_0) \circ s = q'_0 \circ s$$

Trivially ϕ is a CSP morphism, and the lemma holds. □

Putting together both results we get a characterization of refinement in terms of CSP morphism for the class of regular CSP processes.

Theorem 6: Given two regular CSP processes P, and P' the following two conditions are equivalent

- P is a refinement of P'
- There is a CSP morphism from P into P'.

In order to characterize process equality we introduce a stronger notion of CSP morphism. Given two regular processes P and P', a CSP morphism from P into P' is said to be *proper* if the following hold:

1p. If $q * a$ is defined and $\phi(q)$ is defined then $\phi(q) \circ a$ is defined.
2p. $R'(\phi(q)) = R(q)$
3p. $\phi(D) \subseteq D'$

Theorem 7: Given two regular CSP processes P, and P'. There is a proper CSP morphism from P into P' if and only if $P = P'$.

Proof. First suppose that $P = P'$, considering the CSP morphism given in the proof of lemma 5 it is trivial to check that the morphism is proper. Suppose now that there is a proper morphism from P into P'. Using condition (1) of the definition of CSP morphism and condition (1p) of the definition of proper morphism we have $traces(P) = traces(P')$ working in the same way as in the proof of lemma 4 we get $P = P'$. □

We introduce canonical forms for regular processes using general automata minimization techniques [Eil74]. Let us first recall the definition of P/s where s is a trace of P for a given CSP process $P = (A, F, D)$. P/s is defined as $P/s = (A, F/s, D/s)$ where F/s is defined as $\{\underline{tR} \mid st\underline{R} \in F\}$ and $D/s = \{t \mid st \in d\}$.

Given a regular process P we consider the automaton $can(P) =< M(P), R(P), D(P) >$ where:

1. The base automaton $M(P) =< A, Q, *, q_0 >$ has $Q = \{P/s \mid s \in traces(P)\}$. The initial state q_0 is P. The transition function is defined as $(P/s) * a = P/(sa)$ if $sa \in traces(P)$ otherwise is undefined.
2. The refusal mapping is defined as $R(P)(P/s) = refusals(P/s)$.
3. If there is some s in $traces(P)$ such that $P/s = Chaos$ then $D(P) = \{Chaos\}$ (the canonical form has at most one divergent state).

As P is a regular process it is given by a determinstic CSP automaton. The mapping defined by $\phi : P \rightarrow can(P)$ defined by $\phi(q_0 * s) = P/s$ is a proper CSP morphism, thus we have:

Theorem 8: Given a regular process P we have $P = can(P)$.

Thus $can(P)$ is the canonical form of process P. Furthermore this form is unique and has minimum number of states over all other automata recognizing the same process.

4. Rational processes

In this section we consider a new class of CSP processes. This class will be defined in terms of CSP linear systems. We call these processes *rational processes*. CSP linear systems are in many aspects similar to the usual linear systems in automata theory. In linear systems the approach via least fixpoint in a CPO can be changed by the well known Arden's lemma. Let us give a formal definition of CSP linear functions.

Given n process identifiers X_1, X_2, ... X_n over the alphabet A, the class of *linear functions* is defined inductively as:

- $Stop_A$ and $Chaos_A$ are linear.
- X_i and $(a \rightarrow X_i)$ are linear, for all i and for all event $a \in A$.
- Let $F(X_1, \ldots, X_n)$ and $G(X_1, \ldots, X_n)$ be two linear functions, then $F \sqcap G$ and $F \,\Box\, G$ are linear.

Given n linear funcions F_1,F_2,..., F_n over n process identifiers X_1, X_2, ... X_n, a *CSP linear system* is formed by the equations $X_i = F_i(X_1, \ldots, X_n)$ for $i = 1, \ldots, n$. In general we denote a CSP linear system as $\mathbf{X} = \mathbf{F}(\mathbf{X})$. A CSP process will be called *rational* if it can be obtained as a solution of a CSP linear system.

Our first result shows that any CSP linear system can be written in some "standard" way. We ilustrate the procedure with an example. We consider the following system:

$$X = (a \rightarrow X \,\Box\, b \rightarrow Y) \sqcap b \rightarrow Z$$
$$Y = Chaos$$
$$Z = c \rightarrow X \sqcap b \rightarrow Y \sqcap c \rightarrow Y$$

The system can be rewritten as:

$$\begin{aligned} X &= \underline{ac} + aX + bY + bZ \\ Y &= Chaos \\ Z &= \underline{ab} + \underline{ac} + cX + (b+c)Y \end{aligned}$$

Defining the following matrix

$$\mathbf{F} = \begin{pmatrix} a & b & b \\ \{\} & \{\} & \{\} \\ c & (c+b) & \{\} \end{pmatrix}$$

and the vector $\mathbf{R}$

$$\mathbf{R} = \begin{pmatrix} \underline{ac} \\ Chaos \\ \underline{ab} + \underline{ac} \end{pmatrix}$$

the process can be expressed as $\mathbf{X} = \mathbf{F}\,\mathbf{X} + \mathbf{R}$.

The ideas given in this example can be formalized to get the following

Lemma 9: Every CSP linear system can be written as $\mathbf{X} = \mathbf{F}\,\mathbf{X} + \mathbf{R}$ where $\mathbf{F}$ is a matrix formed by subsets of A and $\mathbf{R}$ is a vector formed by subsets of P($\underline{A}$) or *Chaos*.

Recall that given an equation $\mathbf{X} = \mathbf{F}(\mathbf{X})$. The solution of the equation can be obtained as $\mu\mathbf{X}.\mathbf{X} = \sqcup_{i \geq 0}\mathbf{F}^i(\mathbf{Chaos})$. When $\mathbf{X}$ is defined by a linear system, the "fix point" approach can be replaced by the "Arden's lemma" approach. By a simple substitution we get

Proposition 10: Let $\mathbf{X}$ be a CSP linear system defined by the equation $\mathbf{X} = \mathbf{F}\,\mathbf{X} + \mathbf{R}$ then $\mu\mathbf{X}.\mathbf{X} = \mathbf{F}^*\,\mathbf{R}$ where $\mathbf{F}^*$ denotes the reflexive and transitive closure of $\mathbf{F}$.

Looking at the previous example, the reflexive and transitive closure of $\mathbf{F}$ is:

$$\mathbf{F}^* = \begin{pmatrix} (a+bc)^* & (a+bc)^*(b+b^2+bc) & (a+bc)^*b \\ \{\} & <> & \{\} \\ c(a+bc)^* & c(a+bc)^*(b+b^2+bc)+b+c & c(a+bc)^*+<> \end{pmatrix}$$

Finally the processes are:

$$\begin{aligned} X =& (a+bc)^*\underline{ac} + (a+bc)^*(b+b^2+bc)Chaos + (a+bc)^*b(\underline{ab}+\underline{ac}) \\ Y =& Chaos \\ Z =& c(a+bc)^*\underline{ac} + (c(a+bc)^*(b+b^2+bc)+b+c)Chaos \\ & + (c(a+bc)^*+<>)(\underline{ab}+\underline{ac}) \end{aligned}$$

Note that the description of processes X and Z obtained by this method is rather convoluted. It can be proved that:

$$X = a^*(\underline{ac} + b\,Chaos)$$
$$Z = \underline{ab} + \underline{ac} + (b + c)Chaos$$

Finally we prove the equivalence between rational and regular processes.

Theorem 11: The class of rational CSP processes equals the class of regular CSP processes.

Proof. Let X_1 be a process described by a linear system $\mathbf{X} = \mathbf{F}\,\mathbf{X} + \mathbf{R}$. Let us construct a CSP automata $P =< M, R, D >$ with $M =< A, Q, T, Q_0 >$ such that $P = X_1$. The automaton will have an state q_i for each variable X_i and the set Q_0 is $\{q_1\}$. The divergent states correspond to those variables that are just *Chaos*. The refusals mapping is $R(q_i) = \mathit{refusals}(X_i)$. The transition function is defined as follows. When q_i is divergent the only transitions are (q_i, a, q_i) for every $a \in A$. When q_i is nondivergent, the process X_i is $\cup_{1 \leq j \leq n} F_{ij}\,X_j + R_i$, then we add (q_i, a, q_j) to T for every $a \in F_{ij}$ and j. Trivially, the automaton P recognizes X_1.

Let $P =< M, R, D >$ be a regular process, with $M =< A, Q, T, Q_0, >$. We consider a variable X_i for each state q_i. There are four different cases giving every one a different equation.

- When q_i is a divergent state we add the equation $X_i = Chaos$.
- When q_i has no outgoing arcs, we add the equation $X_i = Stop$. Note that in this case $R(q_i) = \underline{A}$.
- When q_i has some outgoing arcs and $R(q_i) = \underline{A}$, for every event a we consider the set $T_a = \{j \mid (q_i, a, q_j) \in T\}$ and the family $\mathcal{T} = \{T_a \mid a \in A \text{ and } T_a \neq \{\}\}$. We define the process
$$Arcs_i = \Box_{T_a \in \mathcal{T}}(\Box_{j \in T_a}(a \rightarrow X_j))$$
Finally, we add the equation $X_i = Arcs_i \sqcap Stop$.
- When q_i has some outgoing arcs and $R(q_i) \neq \underline{A}$. Considering only maximal elements under inclusion, the set of refusals can be written as $R(q_i) = \{\underline{R_{i1}}, \ldots, \underline{R_{ik}}\}$. As every $\underline{R_{i\ell}} \neq \underline{A}$ every $\overline{R}_{i\ell} = \underline{A} \setminus \underline{R_{i\ell}}$ is different from the empty set. For every $a \in \overline{R}_{i\ell}$ we define $\overline{R}^a_{i\ell} = \{j \mid (q_i, a, q_j) \in T\}$. As $\underline{R_{i\ell}}$ is maximal the set $\{a\} \cup \underline{R_{i\ell}}$ is not a refusal. That implies the existence of a state q_j such that $(q_i, a, q_j) \in T$, then $\underline{R^a_{i\ell}}$ is non empty. We consider the following process
$$Refusals_i = \sqcap_\ell(\Box_{a \in \overline{R}_{i\ell}}(\Box_{j \in \overline{R}^a_{i\ell}}(a \rightarrow X_j)))$$
By construction $\mathit{refusals}(Refusals_i) = R(q_i)$. The final equation is $X_i = Arcs_i \sqcap Refusals_i$, where $Arcs_i$ is defined as before.

The final process P is defined as $P = X_1 \sqcap \ldots \sqcap X_p$ where $q_1, \ldots, q_p$ are the initial states.□

5. Closure properties

In this section we study under which CSP operations the class of regular (or rational) processes remains unchanged. By definition CSP linear systems are closed under prefixing and internal and external nondeterminism. Let us now consider other operations.

Lemma 12: Regular CSP processes are closed under concealment.

Proof. Suppose that a regular process P is given by a CSP-automaton. That is P is given by a set of linear equations of the forms described in the proof of theorem 11. We want to find a linear system for $P\backslash b$, for a given $b \in A$. Concealing b in the linear system defining P give us a second system of equations in which we have some unguarded variables. Thus our problem is to find a way to remove such equations. To do this we consider an auxiliar graph, with a node for each variable, such that (i, j) is an arc iff X_j appears as an unguarded variable in the equation defining X_i. When the graph has no cycles, the unguarded equations can be avoided by substitution. When the graph contains a cycle, the least fix point for the variables involved in it is just $Chaos$. Then we can substitute these variables by $Chaos$. Once all cycles are removed all other unguarded variables can be avoided by substitution. Note that equations like $X_i = Chaos \sqcap X_j \ldots$ can appear after substitution, in this case we equal X_i to $Chaos$. Thus $P\backslash b$ is linear.□

Given a regular process P and a one-to-one function $f : A \to B$. It is easy to see that the process $f(P)$ is recognized by the same automaton changing every event a by $f(a)$. Then we have

Lemma 13: Regular CSP process are closed under a change of symbols under one-to-one functions.

Lemma 14: Regular CSP process are closed under interleaving.

Proof. Consider two regular process P' and P''. Let us construct a CSP automaton P recognizing $P' \| P''$. The base automaton has as alphabet the union of alphabets. The states set is the cartesian product. One state $[q', q'']$ is divergent whenever q' or q'' is divergent, for these states the refusal set is the whole alphabet. For nondivergent states the refusal set is $R([q', q'']) = R'(q') + R''(q'')$. The transition function is defined in order to consider the different ways of interleaving. As usual a divergent state loops to itself. When $[q', q'']$ is non-divergent we have two cases. First, when both states evolve using a common event for P' and P'', the automaton needs to evolve to the corresponding pair of states. Second, when one of the states evolves using a noncommon event, the other state remains unchanged. □

Putting together the precedent lemmas we get:

Theorem 15: The class of CSP regular processes is stable by internal nondeterminism, external nondeterminism, change of symbol under one-one functions, prefixing and interleaving.

6. Conclusions and open questions

Our motivations to study CSP-automata were primarly theoretical. Roughly speaking failures semantic seems to be a sophisticated formal language, thus our aim was to provide it with an adequate state device. We have shown that CSP-automata follows closely the semantic of failures. For example a law as $(x \rightarrow P) \sqcap (x \rightarrow Q) = (x \rightarrow (P \sqcap Q))$ is implicit in the determinization algorithm and laws like $P \sqcap P = P$ or $P \sqcap \mathit{Chaos} = \mathit{Chaos}$ is implicit in the minimization routine. By all these facts CSP-automata are an interesting and workable tool. We belive that the notations and techniques of automata and formal languages theory are useful tools in proving properties about processes. This paper is an essay to do it.

In a more practical view, we think that CSP-automata could be used to model some real small processes. Of course we have the problem of the explosion of the number of states, but this will happen with any state based approach. The advantadges of CSP-automata respecting to other models are a careful and economical treatement of nondeterminism and divergences.

As CSP-automata can be a usefull tool, it is important to look at the computational complexity of the problems related to them. For example the equivalence problem in CSP-automata stated as: "verify if two nondeterministic CSP-automata give the same process" is *PSPACE-complete*. As proving equivalence is computationally too hard it would be necessary some alternative approach. An adecuate notion of bisimulation (up and down [Jos88, HeJ89]) matching with CSP-automata could be very useful. As bisimulation equivalence for CCS automata is a *P-complete* problem [Mil89, KaS90, BGS90], we have a strong feeling that bisimulation equivalence between CSP-automata will be a *P-complete* problem.

References

[BGS90] Balcázar, J.L., Gabarró, J., Santha, M.: Deciding Bisimilarity is P-complete, Report LSI-90-25.

[Eil74] Eilenberg, S.: *Automata, Languages and Machines*, vol A. Academic Press, 1974.

[Hoa85] Hoare, C.A.R.: *Communicating Sequential Processes.* Prentice Hall, 1985.

[HeJ89] He Jifeng.: Process Simulation and Refinement, *Formal Aspects of Computing* 1, 229–241 (1989).

[Jos88] Josephs, M.B.: A State-Based Approach to Communicating Processes, *Distributed Computing*, 3, 9–18 (1988).

[Mil89] Milner, R.: *Communication and Concurrency* Prentice Hall, 1989.

[KaS90] Kanellakis, P.C., Smolka, S.A.: CCS Expressions, Finite State Processes, and Three Problems of Equivalence, *Information and Computation*, 86, 43–68 (1990).