Acta Informatica 15, 209-217 (1981)

Acta Informatica

Flow Languages Equal Recursively Enumerable Languages

Toshiro Araki and Nobuki Tokura

Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University, Toyonaka, Osaka 560, Japan

Summary. Recently, A.C. Shaw introduced a new class of expressions called flow expressions, and conjectured that the formal descriptive power of flow expressions lies somewhat below context-sensitive grammers. In this paper, we give a negative answer for his conjecture, that is, we show that all recursively enumerable languages may be denoted by flow expressions.

1. Introduction

Recently, A.C. Shaw [4] introduced a new class of expressions, called flow expressions, which are extended regular expressions to describe concurrencies, synchronization and cyclic activities. Flow languages are defined as languages which are denoted by flow expressions under the restrictions imposed by the lock and wait/signal symbols. The lock symbols handle critical sections and the wait/signal symbols provide a simple synchronization mechanism. As similar non-procedural description languages based on regular expressions, path expressions [1] and event expressions [3] are known. Path expressions are used to describe the synchronization and coordination among processes. Event expressions are strongly related to flow expressions, that is, only one difference between them is the synchronization mechanism.

It is known that any recursively enumerable language is described by some event expression [3]. But as for flow expressions, A.C. Shaw conjectured that the formal descriptive power of flow expressions (excluding the cyclic operator) lies somewhat below context-sensitive grammers and is incomparable with (neither above nor below) context-free grammars. In this paper, however, we give a negative answer to his conjecture. That is, every recursively enumerable language is shown to be a flow language. Since the converse trivially holds, the classes of recursively enumerable languages and flow languages coincide. Since the flow languages are defined as the smallest family of languages containing some very basic ones and closed under a few basic operations, a new characterization of recursively enumerable languages is obtained.

0001-5903/81/0015/0209/$01.80

2. Definitions

In this section, the definitions of flow expressions and flow languages are presented. In [4], a cyclic operator ∞ is introduced, but the following discussion can be done without it. This exclusion of the cyclic operator leads to somewhat simpler discussion because the operator produces infinite strings.

Let Σ^* denote the set of all finite strings composed of symbols of Σ including the empty string λ whose length is zero. For $\Pi \subset \Sigma$, let h_Π be the homomorphism from Σ^* to Π^* such that $h_\Pi(a)=a$ for $a \in \Pi$ and $h_\Pi(a)=\lambda$ for $a \notin \Pi$. For a set Π, let $|\Pi|$ denote the number of elements in Π.

Let Σ be a finite set of atomic symbols, $\Gamma=\{{}_1[\,,\,]_1, {}_2[\,,\,]_2, \ldots, {}_c[\,,\,]_c\}$ be a finite set of lock symbols, and $\Omega=\{\sigma_1, \omega_1, \sigma_2, \omega_2, \ldots, \sigma_d, \omega_d\}$ be a finite set of wait/signal symbols, where $\Sigma \cap \Gamma = \Sigma \cap \Omega = \Gamma \cap \Omega = \{\ \}$[1]. (In examples where $c=1$, we drop the subscripts on symbols in Γ.)

Definition 1. Flow expressions are constructed by the following rules[2].

(1) Each $a \in \Sigma \cup \Omega$, λ, and ϕ are flow expressions.

(2) If S and S' are flow expressions, then (S), SS', $S+S'$, S^*, $S \odot S'$ and $S^{\circledast}$ are flow expressions.

(3) If S is a flow expression and ${}_k[\,,\,]_k$ is a pair of lock symbols in Γ, then ${}_k[S]_k$ is a flow expression.

We define S^i and $S^{\odot i}$ as follows:

(1) $S^0=\lambda$.
(2) $S^i=S^{i-1}S$ for $i>0$.
(3) $S^{\odot 0}=\lambda$.
(4) $S^{\odot i}=S^{\odot i-1} \odot S$ for $i>0$.

Definition 2. The language $\hat{L}(S)$, which imposes no interpretation on the lock and wait/signal symbols, is defined as follows:

(1) $\hat{L}(\phi)=\{\ \}$.
(2) $\hat{L}(\lambda)=\{\lambda\}$.
(3) $\hat{L}(a)=\{a\}$ for $a \in \Sigma \cup \Gamma \cup \Omega$.
(4) $\hat{L}((S))=\hat{L}(S)$.
(5) $\hat{L}(SS')=\{x\,y \mid x \in \hat{L}(S) \text{ and } y \in \hat{L}(S')\}$.
(6) $\hat{L}(S+S')=\{x \mid x \in \hat{L}(S) \text{ or } x \in \hat{L}(S')\}$.
(7) $\hat{L}(S^*)=\bigcup_{i=0}^{\infty} \hat{L}(S^i)$.
(8) $\hat{L}(S \odot S')=\{x_1 y_1 x_2 y_2 \ldots x_k y_k \mid x_1 x_2 \ldots x_k \in \hat{L}(S) \text{ and } y_1 y_2 \ldots y_k \in \hat{L}(S')\}$.
(9) $\hat{L}(S^{\circledast})=\bigcup_{i=0}^{\infty} \hat{L}(S^{\odot i})$.

Definition 3. Flow expressions S_{signal} and S_{lock} are defined as follows:

$$S_{\text{signal}}=(\sigma_1\omega_1+\sigma_1)^* \odot (\sigma_2\omega_2+\sigma_2)^* \odot \ldots \odot (\sigma_d\omega_d+\sigma_d)^*.$$

$$S_{\text{lock}}=({}_1[\ ]_1)^* \odot ({}_2[\ ]_2)^* \odot \ldots \odot ({}_c[\ ]_c)^*.$$

---

[1] The empty set is denoted by $\{\ \}$

[2] In [4], operator $\cup$ is used instead of operator $+$

Definition 4. The language $L(S)$, which is called the flow language defined by S, is obtained from $\hat{L}(S)$ by applying the restrictions imposed by the lock and wait/signal symbols as follows:

$$L(S)=\{x_1 x_2 \dots x_k | z=x_1 y_1 x_2 y_2 \dots x_k y_k \in \hat{L}(S),\ x_i \in \Sigma^*,\ y_i \in (\Omega \cup \Gamma)^*,$$

$$\text{and } y_1 y_2 \dots y_k \in \hat{L}(S_{\text{lock}} \odot S_{\text{signal}})\}, \text{ that is,}$$

$$L(S)=\{h_\Sigma(z) | z \in \hat{L}(S),\ h_\Omega(z) \in \hat{L}(S_{\text{signal}}) \text{ and } h_\Gamma(z) \in \hat{L}(S_{\text{lock}})\}.$$

Though the lock symbols can be simulated with the wait/signal symbols, we use them for simplicity and readability.

Example 1. Let $S_1=(ab)^{\circledast}$, where $\Sigma=\{a,b\}$ and $\Gamma=\Omega=\{\ \}$. Then $L(S_1)=\{w|$ the number of a's is greater than or equal to the number of b's in all prefixes of w, and equal in $w\}$.

Example 2. Let $S_2=((ab)^{\circledast} c)^*$, where $\Sigma=\{a,b,c\}$ and $\Gamma=\Omega=\{\ \}$. Then $L(S_2)=\{w_1 c w_2 c \dots w_k c | w_i \in L(S_1)$ for $1 \leqq i \leqq k\}$, where S_1 is defined in Example 1.

Example 3. Let $S_E=(([\sigma_1][\omega_2])^*([\sigma_3][\omega_4])^*) \odot ([\omega_1 a \sigma_2][\omega_3 b \sigma_4])^{\circledast}$, where $\Sigma=\{a,b\}$, $\Gamma=\{[\ ,\ ]\}$ and $\Omega=\{\sigma_1, \omega_1, \sigma_2, \omega_2, \sigma_3, \omega_3, \sigma_4, \omega_4\}$. Then, $L(S_E)=\{a^n b^n | n \geqq 0\}$.

This can be seen as follows: The string surrounded with lock symbols [and] is treated as atomic or indivisible. Therefore, it is sufficient to consider the subset $\{w \in \{[\omega_1 a \sigma_2], [\omega_3 b \sigma_4]\}^* | w$ contains equal numbers of $[\omega_1 a \sigma_2]$'s and $[\omega_3 b \sigma_4]$'s$\}$ of $\hat{L}(([\omega_1 a \sigma_2][\omega_3 b \sigma_4])^{\circledast})$. The regular expression $([\sigma_1][\omega_2])^*([\sigma_3][\omega_4])^*$ denotes the set of strings with any numbers of repeated $([\sigma_1][\omega_2])$ followed by any numbers of repeated $([\sigma_3][\omega_4])$. $\hat{L}(S_E)$ denotes the set made by shuffling two sets $\hat{L}(([\sigma_1][\omega_2])^*([\sigma_3][\omega_4])^*)$ and $\hat{L}(([\omega_1 a \sigma_2][\omega_3 b \sigma_4])^{\circledast})$. $L(S_E)$ is a set of strings of the form $h_\Sigma(z)$ such that z fulfills the lock and wait/signal restrictions and z is in $\hat{L}(S_E)$. These restrictions allow only strings of the form $([\sigma_1][\omega_1 a \sigma_2][\omega_2])^n([\sigma_3][\omega_3 b \sigma_4][\omega_4])^n$. Thus, $L(S_E)=\{a^n b^n | n \geqq 0\}$.

The following proof will employ essentially the same technique as this example.

3. Proof of the Theorem

In this section, we give the proof of the following theorem.

Theorem. *The flow languages equal the recursively enumerable languages.*

It is trivial that the flow languages are the recursively enumerable languages. Therefore, we show that the recursively enumerable languages are the flow languages, that is, every recursively enumerable language is denoted by some flow expressions. To show this, we consider a deterministic two-counter automaton (abbreviated as $2ca$) with one, one-way read-only input tape. It is known that a $2c\,a$ can simulate a Turing machine, that is, recursively enumerable languages can be recognized by $2ca$'s [2]. Without loss of generality, we have the following definitions and assumptions on $2caK$.

(1) The set of states is denoted by Π.

(2) The two counters are denoted by c_1 and c_2.

(3) The set of input symbols is denoted by Δ.

(4) K starts from the initial state s_0 and halts when and only when it goes to the final state s_f.

(5) In states s_0 and s_f, the contents of c_1 and c_2 are to be zero.

(6) For simplicity, only four types of operations defined below are considered to be used. This does not lose any generality, because any other operation can be simulated by using these operations.

(I) If the current state is s_i, then add one to counter c_l and go to state s_j. This operation is denoted by $(+, s_i, s_j, l)$. Let P be the set of these operations.

(II) If the current state is s_i and the content of counter c_l is not equal to zero, then subtract one from counter c_l and go to state s_j. This operation is denoted by $(-, s_i, s_j, l)$. Let M be the set of these operations.

(III) If the current state is s_i and the content of counter c_l is equal to zero, then go to state s_j. This operation is denoted by $(=, s_i, s_j, l)$. Let Z be the set of these operations.

(IV) If the current state is s_i and the input head scans the input symbol a, then move the input head one cell to the right and go to state s_j. This operation is denoted by $(*, s_i, s_j, a)$. Let R be the set of these operations.

(7) In state s_i, if some operation in M is possible with counter l, then another operation in Z is also possible with counter l, and vice versa.

(8) The last operation is in Z for both counters.

A configuration of the $2caK$ is represented by $\langle s, k_1, k_2, w\rangle$ where s is the current state of K, k_1 and k_2 are the current contents of counters c_1 and c_2 of K, respectively, and w is the input tape which is still to be scanned. The initial and final configurations are $\langle s_0, 0, 0, w\rangle$ and $\langle s_f, 0, 0, \lambda\rangle$, respectively. An expression $\langle s, k_1, k_2, ww'\rangle \overset{\alpha}{\underset{K}{\vdash\!\!-}} \langle s', k'_1, k'_2, w'\rangle$ or $\langle s, k_1, k_2, ww'\rangle \overset{\alpha}{\vdash\!\!-} \langle s', k'_1, k'_2, w'\rangle$ if K is known will denote that the $2caK$ will reach the configuration $\langle s', k'_1, k'_2, w'\rangle$ through an operation sequence α starting from the configuration $\langle s, k_1, k_2, ww'\rangle$ and scanning the portion w of the input tape. Let $L_0(K)$ be the set of input tapes w such that $\langle s_0, 0, 0, w\rangle \overset{\alpha}{\vdash\!\!-} \langle s_f, 0, 0, \lambda\rangle$ for some α.

Now we shall show the method of constructing flow expressions which correspond to $2ca$'s.

Definition 5. Given a $2caK$, the flow expressions S_K and S_0 are constructed as follows:

(1) Let $\Sigma = \Delta$, $\Gamma = \{[\,,\,]\}$, and $\Omega = \Omega_\Pi \cup \Omega_C$ where $\Omega_\Pi = \{\sigma_i, \omega_i | s_i \in \Pi\}$ and

$$\Omega_C = \{\sigma_{lpj}, \omega_{lpj}, \sigma_{lmj}, \omega_{lmj}, \sigma_{lzj}, \omega_{lzj} \mid 1 \leqq l \leqq 2, 1 \leqq j \leqq 2\}.$$

(2) For each operation $P_i = (+, s_q, s_r, l) \in P$ in K, let

$$S_{P_i} = [\omega_q][\sigma_{lp1}][\omega_{lp2}][\sigma_r] \quad (1 \leqq i \leqq |P|).$$

(3) For each operation $M_i=(-s_q, s_r, l)\in M$ in K, let

$$S_{M_i}=[\omega_q][\sigma_{lm1}][\omega_{lm2}][\sigma_r] \quad (1\leqq i\leqq |M|).$$

(4) For each operation $Z_i=(=, s_q, s_r, l)\in Z$ in K, let

$$S_{Z_i}=[\omega_q][\sigma_{lz1}][\omega_{lz2}][\sigma_r] \quad (1\leqq i\leqq |Z|).$$

(5) For each operation $R_i=(*, s_q, s_r, a)\in R$ in K $(a\in \Delta)$, let

$$S_{R_i}=[\omega_q][a][\sigma_r] \quad (1\leqq i\leqq |R|).$$

(6) Let

$$S_0=(([\omega_{1p1}\sigma_{1p2}][\omega_{1m1}\sigma_{1m2}])^{\circledast}[\omega_{1z1}\sigma_{1z2}])^* \odot(([\omega_{2p1}\sigma_{2p2}][\omega_{2m1}\sigma_{2m2}])^{\circledast}[\omega_{2z1}\sigma_{2z2}])^*.$$

(7) Let

$$S_K=[\sigma_0](S_{P_1}+S_{P_2}+\ldots+S_{P_{|P|}}+S_{M_1}+S_{M_2}+\ldots+S_{M_{|M|}}+S_{Z_1}+S_{Z_2}+\ldots+S_{Z_{|Z|}} +S_{R_1}+S_{R_2}+\ldots+S_{R_{|R|}})^*[\omega_f]\odot S_0.$$

In the method of constructing S_K, we use the technique used in Example 3 to compare to the numbers of a's and b's. That is, σ_{lp1}, ω_{lp2}, σ_{lm1}, ω_{lm2}, ω_{lp1}, σ_{lp2}, ω_{lm1} and σ_{lm2} in S_K correspond to σ_1, ω_2, σ_3, ω_4, ω_1, σ_2, ω_3 and σ_4 in Example 3, respectively. By using this technique, the property that the number of operations in P is always greater than or equal to that in M and always exactly equal before the operations in Z for a counter c_l in the $2caK$ is embedded in flow expression S_K. Furthermore, lock symbols [and] are used to forbid any strings to be shuffled between ω_{lt1} and σ_{lt2} in S_0 for $l=1$ or 2 and $t=p$ or m or z. Our goal is to show that $L(S_K)=L_0(K)$.

For simplicity, we use the notation $\bar{\sigma}$, $\bar{\omega}$ and $\bar{a}$ to denote $[\sigma]$, $[\omega]$ and $[a]$, respectively.

Definition 6. For a finite operation sequence α in the 2caK, define $f(\alpha)$ and $g(\alpha)$ recursively as follows:

(1) If $\alpha=\lambda$, then

$$f(\alpha)=g(\alpha)=\lambda.$$

(2) If $\alpha=P_i\alpha'$ and $P_i=(+, s_q, s_r, l)\in P$, then

$$f(\alpha)=\bar{\omega}_q\bar{\sigma}_{lp1}\bar{\omega}_{lp2}\bar{\sigma}_r f(\alpha') \text{ and}$$
$$g(\alpha)=\bar{\omega}_q\bar{\sigma}_{lp1}[\omega_{lpl}\sigma_{lp2}]\bar{\omega}_{lp2}\bar{\sigma}_r g(\alpha').$$

(3) If $\alpha=M_i\alpha'$ and $M_i=(-, s_q, s_r, l)\in M$, then

$$f(\alpha)=\bar{\omega}_q\bar{\sigma}_{lm1}\bar{\omega}_{lm2}\bar{\sigma}_r f(\alpha') \text{ and}$$
$$g(\alpha)=\bar{\omega}_q\bar{\sigma}_{lm1}[\omega_{lm1}\sigma_{lm2}]\bar{\omega}_{lm2}\bar{\sigma}_r g(\alpha').$$

(4) If $\alpha=Z_i\alpha'$ and $Z_i=(=, s_q, s_r, l)\in Z$, then

$$f(\alpha)=\bar{\omega}_q\bar{\sigma}_{lz1}\bar{\omega}_{lz2}\bar{\sigma}_r f(\alpha') \text{ and}$$
$$g(\alpha)=\bar{\omega}_q\bar{\sigma}_{lz1}[\omega_{lz1}\sigma_{lz2}]\bar{\omega}_{lz2}\bar{\sigma}_r g(\alpha').$$

(5) If $\alpha=R_i\alpha'$ and $R_i=(*,s_q,s_r,a)\in R$, then

$$f(\alpha)=\bar{\omega}_q\bar{a}\bar{\sigma}_r f(\alpha') \text{ and}$$
$$g(\alpha)=\bar{\omega}_q\bar{a}\bar{\sigma}_r g(\alpha').$$

Lemma 1. *Let* $S_k(s_i,k_1,k_2,\alpha)=(\bar{\sigma}_{1p1}\bar{\omega}_{1p2})^{k_1}(\bar{\sigma}_{2p1}\bar{\omega}_{2p2})^{k_2}\bar{\sigma}_i f(\alpha)\bar{\omega}_f\odot S_0$. *If* $\langle s_i,k_1,k_2,w\rangle \vdash\!\frac{\alpha}{K}\!- \langle s_f,0,0,\lambda\rangle$, *then*

$$z=(\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2})^{k_1}(\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}\bar{\sigma}_i g(\alpha)\bar{\omega}_f\in\hat{L}(S_k(s_i,k_1,k_2,\alpha))$$

and $w=h_\Sigma(z)\in L(S_k(s_i,k_1,k_2,\alpha))$.

Proof. We prove this lemma by induction on n where n is the length of α. Consider the case of $n=0$. Since $i=f$, $k_1=k_2=0$ and $\alpha=\lambda$, that is, $\langle s_f,0,0,\lambda\rangle\vdash\!\frac{\lambda}{}\!-\langle s_f,0,0,\lambda\rangle$, $z=\bar{\sigma}_f\bar{\omega}_f$ and $S_k(s_f,0,0,\lambda)=\bar{\sigma}_f\bar{\omega}_f\odot S_0$. Since $\lambda\in\hat{L}(S_0)$, $z=\bar{\sigma}_f\bar{\omega}_f\in\hat{L}(S_k(s_f,0,0,\lambda))$. Since $z\in\hat{L}(S_{\text{signal}}\odot S_{\text{lock}})$, $\lambda=h_\Sigma(z)\in L(S_k(s_f,0,0,\lambda))$. Therefore this lemma is true for $n=0$.

Suppose that this lemma is true for $n=l-1$ and consider the case of $n=l$. Let $\langle s_i,k_1',k_2',w'\rangle\vdash\!\frac{t}{}\!-\langle s_j,k_1,k_2,w\rangle\vdash\!\frac{\alpha}{}\!-\langle s_f,0,0,\lambda\rangle$ where $\alpha'=t\alpha$ and α are two operation sequences of length l and $l-1$, respectively. Assume that

$$z=(\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2})^{k_1}(\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}\bar{\sigma}_j g(\alpha)\bar{\omega}_f\in\hat{L}(S_k(s_j,k_1,k_2,\alpha))$$

and $w=h_\Sigma(z)\in L(S_k(s_j,k_1,k_2,\alpha))$ by the induction hypothesis. Therefore

$$w=h_\Sigma(z)=h_\Sigma(\bar{\sigma}_j g(\alpha)\bar{\omega}_f) \quad\text{and}\quad h_{\Omega\cup\Gamma}(\bar{\sigma}_j g(\alpha)\bar{\omega}_f)\in\hat{L}(S_{\text{signal}}\odot S_{\text{lock}}).$$

In this proof, let $C(k_1,k_2)$ denote

$$(\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2})^{k_1}(\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}.$$

There are four cases with respect to the type of operation t. The discussions below are done for the counter c_1, but they hold for the counter c_2.

(1) If t is in P, that is, $t=(+,s_i,s_j,1)$, then $k_1'=k_1-1$, $k_2'=k_2$ and $w'=w$. Let

$$\begin{aligned}z'&=C(k_1',k_2')\bar{\sigma}_i g(\alpha')\bar{\omega}_f=C(k_1-1,k_2)\bar{\sigma}_i g(\alpha')\bar{\omega}_f=C(k_1-1,k_2)\bar{\sigma}_i g(t)g(\alpha)\bar{\omega}_f\\&=(\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2})^{k_1-1}(\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}\\&\quad\cdot\bar{\sigma}_i\bar{\omega}_i\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2}\bar{\sigma}_j g(\alpha)\bar{\omega}_f.\end{aligned}$$

By the fact that $z\in\hat{L}(S_k(s_j,k_1,k_2,\alpha))$ and the definition of S_0, $z'\in\hat{L}(S_k(s_i,k_1',k_2',\alpha'))$. Since $w'=w=h_\Sigma(\bar{\sigma}_j g(\alpha)\bar{\omega}_f)$ and $h_{\Omega\cup\Gamma}(\bar{\sigma}_j g(\alpha)\bar{\omega}_f)\in\hat{L}(S_{\text{signal}}\odot S_{\text{lock}})$, $h_\Sigma(z')=w'$ and $h_{\Omega\cup\Gamma}(z')\in\hat{L}(S_{\text{signal}}\odot S_{\text{lock}})$. Therefore $w'=h_\Sigma(z')\in L(S_k(s_i,k_1',k_2',\alpha'))$.

(2) If t is in M, that is, $t=(-,s_i,s_j,1)$, then $k_1'=k_1+1$, $k_2'=k_2$ and $w'=w$. Let

$$\begin{aligned}z'&=C(k_1',k_2')\bar{\sigma}_i g(\alpha')\bar{\omega}_f=C(k_1+1,k_2)\bar{\sigma}_i g(\alpha')\bar{\omega}_f=C(k_1+1,k_2)\bar{\sigma}_i g(t)g(\alpha)\bar{\omega}_f\\&=(\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2})^{k_1+1}(\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}\\&\quad\cdot\bar{\sigma}_i\bar{\omega}_i\bar{\sigma}_{1m1}[\omega_{1m1}\sigma_{1m2}]\bar{\omega}_{1m2}\bar{\sigma}_j g(\alpha)\bar{\omega}_f.\end{aligned}$$

By the fact that $z \in \hat{L}(S_k(s_j, k_1, k_2, \alpha))$ and the definition of $S_0, z' \in \hat{L}(S_k(s_i, k'_1, k'_2, \alpha'))$. Since $w' = w = h_\Sigma(\bar{\sigma}_j g(\alpha)\bar{\omega}_f)$ and $h_{\Omega \cup \Gamma}(\bar{\sigma}_j g(\alpha)\bar{\omega}_f) \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$, $h_\Sigma(z') = w'$ and $h_{\Omega \cup \Gamma}(z') \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$. Therefore $w' = h_\Sigma(z') \in L(S_k(s_i, k'_1, k'_2, \alpha'))$.

(3) If t is in Z, that is, $t = (=, s_i, s_j, 1)$, then $k'_1 = k_1 = 0$, $k'_2 = k_2$ and $w' = w$. Let

$$\begin{aligned} z' &= C(k'_1, k'_2)\bar{\sigma}_i g(\alpha')\bar{\omega}_f = C(0, k_2)\bar{\sigma}_i g(\alpha')\bar{\omega}_f = C(0, k_2)\bar{\sigma}_i g(t) g(\alpha)\bar{\omega}_f \\ &= (\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}\bar{\sigma}_i\bar{\omega}_i\bar{\sigma}_{1z1}[\omega_{1z1}\sigma_{1z2}]\bar{\omega}_{1z2}\bar{\sigma}_j g(\alpha)\bar{\omega}_f. \end{aligned}$$

By the fact that $z \in \hat{L}(S_k(s_j, 0, k_2, \alpha))$ and the definition of $S_0, z' \in \hat{L}(S_k(s_i, 0, k_2, \alpha'))$. Since $w' = w = h_\Sigma(\bar{\sigma}_j g(\alpha)\bar{\omega}_f)$ and $h_{\Omega \cup \Gamma}(\bar{\sigma}_j g(\alpha)\bar{\omega}_f) \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$, $h_\Sigma(z') = w'$ and $h_{\Omega \cup \Gamma}(z') \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$. Therefore $w' = h_\Sigma(z') \in L(S_k(s_i, 0, k'_2, \alpha'))$.

(4) If t is in R, that is, $t = (*, s_i, s_j, a)$, then $k'_1 = k_1$, $k'_2 = k_2$ and $w' = a\,w$. Let

$$\begin{aligned} z' &= C(k'_1, k'_2)\bar{\sigma}_i g(\alpha')\bar{\omega}_f = C(k_1, k_2)\bar{\sigma}_i g(\alpha')\bar{\omega}_f = C(k_1, k_2)\bar{\sigma}_i g(t) g(\alpha)\bar{\omega}_f \\ &= (\bar{\sigma}_{1p1}[\omega_{1p1}\sigma_{1p2}]\bar{\omega}_{1p2})^{k_1}(\bar{\sigma}_{2p1}[\omega_{2p1}\sigma_{2p2}]\bar{\omega}_{2p2})^{k_2}\bar{\sigma}_i\bar{\omega}_i\bar{a}\,\bar{\sigma}_j g(\alpha)\bar{\omega}_f. \end{aligned}$$

By the fact that $z \in \hat{L}(S_k(s_j, k_1, k_2, \alpha))$ and $f(t) = \bar{\omega}_i\bar{a}\,\bar{\sigma}_j$, $z' \in \hat{L}(S_k(s_i, k'_1, k'_2, \alpha'))$. Since $w = h_\Sigma(\bar{\sigma}_j g(\alpha)\bar{\omega}_f)$ and $h_{\Omega \cup \Gamma}(\bar{\sigma}_j g(\alpha)\bar{\omega}_f) \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$, $h_\Sigma(z') = a\,w = w'$ and $h_{\Omega \cup \Gamma}(z') \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$. Therefore $w' = h_\Sigma(z') \in L(S_k(s_i, k'_1, k'_2, \alpha'))$. (Q.E.D.)

Lemma 2. *If* $w \in L_0(K)$, *then* $w \in L(S_K)$.

Proof. If $w \in L_0(K)$, then there is a finite operation sequence α such that $\langle s_0, 0, 0, w\rangle \overset{\alpha}{\underset{K}{\vdash}} \langle s_f, 0, 0, \lambda\rangle$. By Lemma 1, $z = \bar{\sigma}_0 g(\alpha)\bar{\omega}_f \in \hat{L}(S_k(s_0, 0, 0, \alpha))$ and $w = h_\Sigma(z) \in L(S_k(s_0, 0, 0, \alpha))$. Since $f(\alpha) \in \hat{L}((S_{P_1} + S_{P_2} + \ldots + S_{P_{|P|}} + S_{M_1} + S_{M_2} + \ldots + S_{M_{|M|}} + S_{Z_1} + S_{Z_2} + \ldots + S_{Z_{|Z|}} + S_{R_1} + S_{R_2} + \ldots + S_{R_{|R|}})^*)$, $\hat{L}(S_k(s_0, 0, 0, \alpha)) \subseteq \hat{L}(S_K)$. Therefore $z \in \hat{L}(S_K)$ and $w = h_\Sigma(z) \in L(S_K)$. (Q.E.D.)

Lemma 3. *If* $w \in L(S_K)$, *then* $w \in L_0(K)$.

Proof. (1) If $w \in L(S_K)$, then there exists at least one string $z \in \hat{L}(S_K)$ such that $h_\Sigma(z) = w$ and $h_{\Omega \cup \Gamma}(z) \in \hat{L}(S_{\text{signal}} \odot S_{\text{lock}})$, and there exists an operation sequence α such that $z \in \hat{L}(\bar{\sigma}_0 f(\alpha)\bar{\omega}_f \odot S_0)$ by the definition of S_K. Since $h_{\Omega_\Pi}(z) \in \hat{L}(S_{\text{signal}, \Omega_\Pi})$ where $S_{\text{signal}, \Omega_\Pi}$ is a flow expression S_{signal} on Ω_Π, that is,

$$\begin{aligned} S_{\text{signal}, \Omega_\Pi} = {} & (\sigma_0\omega_0 + \sigma_0)^* \odot (\sigma_1\omega_1 + \sigma_1)^* \odot (\sigma_2\omega_2 + \sigma_2)^* \\ & \odot \ldots \odot (\sigma_{|\Pi|-2}\omega_{|\Pi|-2} + \sigma_{|\Pi|-2})^* \odot (\sigma_f\omega_f + \sigma_f)^*, \end{aligned}$$

by the definitions of $f(\alpha)$ and S_0, $h_{\Omega_\Pi}(z)$ should be of the form $\sigma_0\omega_0\sigma_{i_1}\omega_{i_1}\sigma_{i_2}\omega_{i_2}\ldots\sigma_{i_l}\omega_{i_l}\sigma_f\omega_f$ for some $i_1, i_2, \ldots, i_l$. Therefore, α is a partially valid operation sequence of K with a state sequence $s_0 s_{i_1} s_{i2} \ldots s_{i_l} s_f$. Here, 'partially valid' means that the sequence α is possible or valid with respect to finite-state part of K but not necessarily possible if the counters are considered. The next step is to show that the operation sequence α is valid, that is, its counter operations do not contain any inconsistency.

(2) By the definition of S_0, for any string $z'(\neq\lambda)$ in $\hat{L}(S_0)$ satisfying the lock conditions, $h_{\Gamma\cup\Omega_{Cl}}(z')$ should be in the set

$$\hat{L}((([\omega_{lp1}\,\sigma_{lp2}]^{u_1}\odot[\omega_{lm1}\,\sigma_{lm2}]^{u_1})[\omega_{lz1}\,\sigma_{lz2}]([\omega_{lp1}\,\sigma_{lp2}]^{u_2}\odot[\omega_{lm1}\,\sigma_{lm2}]^{u_2})[\omega_{lz1}\,\sigma_{lz2}] \ldots([\omega_{lp1}\,\sigma_{lp2}]^{u_r}\odot[\omega_{lm1}\,\sigma_{lm2}]^{u_r})[\omega_{lz1}\,\sigma_{lz2}])\odot S_{\text{lock}})$$

for some $u_1, u_2, \ldots, u_r$, where $\Omega_{Cl}=\{\sigma_{lpj}, \omega_{lpj}, \sigma_{lmj}, \omega_{lmj}, \sigma_{lzj}, \omega_{lzj} | 1\leqq j\leqq 2\}\subset\Omega_C$. It should be noted that there are equal numbers of $[\omega_{lp1}\,\sigma_{lp2}]$ and $[\omega_{lm1}\,\sigma_{lm2}]$ in the prefix preceding any $[\omega_{lz1}\,\sigma_{lz2}]$. This is necessary to ensure that if K executes an operation Z_i in Z for the counter c_l, then the contents of the counter c_l is always zero. Now, since $h_{\Omega\cup\Gamma}(z)\in\hat{L}(S_{\text{signal}}\odot S_{\text{lock}})$, by the fact above and the definition of $f(\alpha)$, $h_{\Gamma\cup\Omega_{Cl}}(z)$ should be in the set $\hat{L}(((p_l^{u_1}\odot m_l^{u_1})\,z_l(p_l^{u_2}\odot m_l^{u_2})\,z_l\ldots(p_l^{u_r}\odot m_l^{u_r})\,z_l)\odot S_{\text{lock}})$ for some $u_1, u_2, \ldots, u_r$, where p_l, m_l and z_l stand for $[\sigma_{lp1}][\omega_{lp1}\,\sigma_{lp2}][\omega_{lp2}]$, $[\sigma_{lm1}][\omega_{lm1}\,\sigma_{lm2}][\omega_{lm2}]$ and $[\sigma_{lz1}][\omega_{lz1}\,\sigma_{lz2}][\omega_{lz2}]$, respectively.

(3) Since $z\in\hat{L}(\bar{\sigma}_0 f(\alpha)\bar{\omega}_f\odot S_0)$ by (1), let $z'\in\hat{L}(\bar{\sigma}_0 f(\alpha)\bar{\omega}_f)$ such that $z=z'\odot z''$ and $z''\in\hat{L}(S_0)$. Then $z'\in\hat{L}(\bar{\sigma}_0(S_{P_1}+S_{P_2}+\ldots+S_{P_{|P|}}+S_{M_1}+S_{M_2}+\ldots+S_{M_{|M|}}+S_{Z_1}+S_{Z_2}+\ldots+S_{Z_{|Z|}}+S_{R_1}+S_{R_2}+\ldots+S_{R_{|R|}})^*\,\bar{\omega}_f)$. Now S_{P_j}, S_{M_j} and S_{Z_j} containing $\sigma_{1p1}, \sigma_{1m1}$ and σ_{1z1} (or $\sigma_{2p1}, \sigma_{2m1}, \sigma_{2z1}$) respectively; by (2) and the definition of S_0, the number of $S_{P_j}(1\leqq j\leqq|P|)$ is equal to that of $S_{M_j}(1\leqq j\leqq|M|)$ in every prefix z''' of z' ending with $S_{Z_j}(1\leqq j\leqq|Z|)$; the number of $S_{P_j}(1\leqq j\leqq|P|)$ is equal to or greater than that of $S_{M_j}(1\leqq j\leqq|M|)$ in every prefix of z'''; and the last symbol of z' is in Z. This implies that in the sequence α, (i) the number of operations in P for the counter c_l is equal to that of operations in M for the counter c_l in every prefix α' ending with an operation in Z for the counter c_l, (ii) in every prefix of α', the number of operations in P is equal to or greater than that of operations in M, and (iii) in the last configuration, the contents of both counters are to be zero. That is, all operations for counters are valid.

(4) By (1) and (3), α is a valid operation sequence of K such that $\langle s_0, 0, 0, h_\Sigma(z)\rangle \overset{\alpha}{\underset{K}{\vdash}} \langle s_f, 0, 0, \lambda\rangle$. Therefore $w=h_\Sigma(z)\in L_0(K)$. (Q.E.D.)

It follows from Lemmas 2 and 3 that $L(S_K)=L_0(K)$. That is, for any $2caK$, there is a flow expression S_K such that $L(S_K)=L_0(K)$. Therefore the proof of Theorem is completed.

4. Conclusion

In this paper, we have shown that the flow languages equal the recursively enumerable languages. Therefore A.C. Shaw's conjecture is resolved negatively. As the result of this fact, almost every decision problem for flow expressions such as the emptiness problem, the equivalence problem and the membership problem is undecidable in general. This answers the decidability problems posed by A.C. Shaw and shows that many questions of practical interest (e.g., deadlock, starvation, verification and correctness) are undecidable for flow expressions. Furthermore, the class of flow languages is closed under almost every operation except for complementation.

References

1. Campbell, R.H., Habermann, A.N.: The specification of process synchronization by path expressions. Lecture Notes in Computer Science, Vol. 16, pp. 89–102. Berlin-Heidelberg-New York: Springer, 1974
2. Minsky, M.L.: Computation: Finite and infinite machines, Englewood Cliffs, N.J.: Prentice-Hall, 1967
3. Ogden, W.F., Riddle, W.E., Rounds, W.C.: Complexity of expressions allowing concurrency. The Fifth Annual ACM Symposium on Principles of Programming Languages 185–194 (1978)
4. Shaw, A.C.: Software descriptions with flow expressions. IEEE Trans. Software Engrg., **SE-4**, 242–254 (1978)

Received June 11, 1979/February 5, 1981