ON THE DESCRIPTIONAL COMPLEXITY OF PRODUCTIOIN SYSTEMS

Peter Trum
Battelle-Institut e.V.
Am Roemerhof 35
D-6000 Frankfurt am Main
West-Germany

Abstract

In this paper we formalize different methods for describing control knowledge in production systems by the concept of production system schemes. In this framework these methods are compared from the viewpoint of descriptional complexity giving us some insight by which means problems can be described in an easy and succinct way.

1 Introduction

In (Georgeff, 82) a production system architecture is described in which procedural knowledge is specified by control languages. In this framework he outlined transformations on production systems to remove the amount of nondetermism to get a more efficient execution. In order to clearly separate the means for describing control knowledge from the (object-level) semantics of a production system and to compare different methods to describe controlknowledge we will abstract this concept by production system schemes. Different methods to describe control information in production systems can then be modelled by different classes of production system schemes (this is in parallel to the theory of program schemes (Greibach, 75)). The concept of production system schemes might also form the basis for a theory of transformations on production systems. In this paper we consider the following classes of production system schemes: controlled production system schemes in which procedural knowledge is described by languages and production system schemes with markers, which use special database symbols to sequence the order of productions. These classes of production system schemes will be compared from the viewpoint of descriptional complexity. We show that removing nondeterminism (if possible) in controlled production systems might result in a size explosion of the equivalent deterministic production system. In the same way we show that in some cases the specification of control-knowledge by markers can result in drastic more succinct representations than a separate description of controlknowledge in even non-deterministic controlled production systems•

2 Notations and Definitions

In the following definitions we introduce the concept of (controlled) production system schemes and production system schemes with markers.

Def.:
A production system schema (PSS) is a quadrupel $Q = (\Sigma, P, A, f)$ with Σ a finite set of production symbols, A a finite set of action symbols, and f a function from Σ into $(P \cup \overline{P})^* \times A$ with $\overline{P} = \{\overline{p} \mid p \in P\}$ denoting the set of negative predicate symbols. For s in Σ $f(s) = (c,a)$ is denoted by $c \longrightarrow a$ where c is the condition (to be interpreted as the conjunction of its elements) and a the action of s.

Def. t
An interpretation of a PSS Q « (Σ, P, A, f) is a pair I = (D, h) where D is a nonempty set, the database of Q, and h is a mapping assigning to each pEP a predicate h(p) from D into (0,1) and to each a $\in$ A a relation over D.

Def .:
A controlled production system schema (CPSS) is a pair Q - (L, P) where P is a PSS and L a language over the production symbols of P.

Def.:
Let $Q = (\Sigma, P, A, f)$ be a PSS, $C = (L, Q)$ be a CPSS and $I = (D, h)$ an interpretation. The execution relation $\Rightarrow$ over $(N_L x D)^2$ where N_L denotes the set of all prefixes of words in L is defined as follows:
$\forall s \in \Sigma$, $\forall x_1, x_2 \in D^2$: $(u, x_1) \Rightarrow (us, x_2)$ iff $f(s) = q \to a$, $h(q)(x_1) = 1$ and $(x_1, x_2) \in h(a)$. The relation I(C) computed by C under I is defined as $I(C) = \{(x, y) / (\varepsilon, x) \overset{*}{\Rightarrow} (w, y), w \in L\}$.
Let Q_1 and Q_2 be two CPSS's. Then Q_1 and Q_2 are equivalent iff $I(Q_1) = I(Q_2)$ under every interpretation I.

The above definitions describe how the execution is controlled by a language. Another method to specify control in production systems consists of allowing the productions to assert certain markers into the data base that then can be tested by other productions. This can be formalized as follows:

Def.:
Let $M = \{v_1, \ldots, v_n\}$ be a set of markers and $V = \{m_1, \ldots, m_k\}$ be a set of values. For $v_i \in M$ and $m_j \in V$ we'll denote by $v_i = m_j$ a new predicate symbol with negation $v_i \neq m_j$ and by $v_i \leftarrow m_j$ a new action symbol with the fixed meaning: "assert that vi has value mj into the database".

Def.:
A production system schema with markers (MPSS) is a tuple $Q = (\Sigma, P, A, M, V, f)$ with Σ, P, A defined as for production system schemas, M a set of markers, V a set of values for markers in M and f a function from Σ into the set $(P \cup \bar{P} \cup \{v = m_j, v_i \neq m_j / v_i \in M, m_j \in V\})^* x (A \cup \{v_i \leftarrow m_j / v_i \in M, m_j \in V\})^*$.

The definition of execution under an interpretation I = (D, h) must be modified in the obvious way so that the meanings of the new predicates and actions are fixed (see e.g. Engelfriet, 74).

In the following we will restrict ourselves to regular control languages. In order to compare the sizes of production systems we next introduce our size measures. Clearly the size of a PSS P is given by the number of its productions. In the case of a CPSS we also have to measure the control language complexity. This can be done by the size of the automaton accepting this language. For a regular control language L the number of states in the nondeterministic (NPA) or deterministic (DFA) finite automaton M accepting L then is a good measure for the descriptional complexity of L (in this case L is also denoted by T(M)).

3 Main Results

If only one production application at each execution step of a CPSS Q = (L, P) can eventually lead to a successful termination, then it's possible to translate it into an equivalent deterministic production system. If in addition the control language is regular we can transform the production system into an equivalent flowchart program.

It is easy to show that in such cases the number of states in the DFA M accepting the language s(L) gives us the size-complexity (i. e. number of statements) of the equivalent flowchart program where s denotes the following substitution from the production symbols into the predicate and action symbols of P:

s(p) = qf where
f(p) = q -->f

Theorem 1

For all n>l there exists a CPSS P_n with size complexity 0(n) s. t. every equivalent flowchart program Q_n has a size complexity of at least 2^n.

Proof:

Let P_n be defined as follows:

$P_n = (L_n, Q_n)$ where
$Q_n = (\{p_1, p_2, p_3\}, \{p, q\}, \{g_1, g_2\}, f)$ with:
$f(p_1) = p\underline{q} \to g_1$,
$f(p_2) = \underline{pq} \to g_2$,
$f(p_3) = \overline{pq} \to nil$,
$L_n = \{p_1, p_2\}^* \cdot \{p_1\} \cdot \{p_1, p_2\}^{n-1} \cdot \{p_3\}$

Now it's easy to see that P_n allows for deterministic execution and that

$s(L_n) = \{pqg_1, p\bar{q}g_2\} \cdot \{pqg_1\} \cdot \{pqg_1, p\overline{qg}_2\}^{n-1} \{\overline{pq}\}$

Since Ln can be accepted by a NFA with 0(n) states it also follows that the (nondeterministic) description of P_n has a size complexity of 0(n). On the other hand it can be shown (Trum, Wotschke, 83) that each DFA M with T(M) = s(Ln) needs at least 2^n states prooving our theorem.

If we consider a MPSS Q with a markerset M the execution order of productions is only constrained by the testing/setting of markers (i. e. the control language is Σ^*). But, as shown in (Engelfriet, 74), the constraints introduced by the use of markers can be modelled by a regular controllanguage CM. This means that in such cases the control language is defined in the productions themselves. This leads us to the following characterisation for the equivalence of MPSS's and CPSS's:

Theorem 2

Let Q1 be a MPSS with markers in M and Q2 = (L, P) be a CPSS. Then Q1 and Q2 are equivalent if $s(L) = h(s(C_M))$ where h is the following substitution: $h(v_i = m_j) = h(v_i = m_j) = h(v_i \leftarrow m_j) = \varepsilon$ and $n(a) = a$ otherwise.

This result leads us to our last theorem:

Theorem 3

For all $n \geq 1$ there exists a MPSS P_n with size complexity 0(n) s. t. every equivalent CPSS C_n has a size complexity of at least 2^n.

Proof

Let $P_n = (\Sigma, P, A, M, V, f)$ with $P = \{p\}$, $A = \{g_1, g_2\}$, $M = \{q, v_1, \ldots, v_n\}$, $V = \{1, \ldots 2n+1\}$, $\Sigma = \{p_1, \ldots, p_{2n+1}, \bar{p}_1, \ldots, \bar{p}_{2n}\}$ and f defined as follows:

$f(p_i) = (q = i \wedge p \rightarrow q \leftarrow i+1 \wedge v_i \leftarrow 1 \wedge g_1)$

$f(\bar{p}_i) = (q = i \wedge \bar{p} \rightarrow q \leftarrow i+1 \wedge v_i \leftarrow 0 \wedge g_2)$

$f(p_{n+i}) = (q = n+i \wedge v_i = 1 \rightarrow q \leftarrow n+i+1 \wedge g_1)$

$f(\bar{p}_{n+i}) = (q = n+i \wedge v_i = 0 \rightarrow q \leftarrow n+i+1 \wedge g_2)$

for $1 \leq i \leq n$ and

$f(p_{2n+1}) = (q = 2n+1 \rightarrow nil)$.

Furthermore, the initial value for the markers is 1 and the goal state is any state where q has value 2n+l.

Since we do not allow to introduce new predicate or function symbols, any equivalent CPSS $C_n = (L_n, Q_n)$ without any markers can have only productions of the form $c \rightarrow a$ where $c \in \{p, \bar{p}\}$ and $a \in \{g_1, g_2\}$

Because of theorem 2 these productions have to be sequenced by the language L_n s.t. $s(L_n) = \{w_1 w_2 / w_1 \in \{pg_1, pg_2\}^n, w_2 \in \{g_1, g_2\}^n$ s.t. $h(w_1) = w_2$ where

$h(p) = h(\bar{p}) = \varepsilon$, $h(g_1) = g_1$ and $h(g_2) = g_2\} = \mathcal{L}_n$.

For any language L_n with $s(L_n) = \mathcal{L}_n$ it holds that any pushdown automaton accepting L_n needs 2^n state or stacksymbols (Meinecke-Schmidt, 78) no matter how the productions in Q_n look like. From this it follows that the specification of the control language by even more general context - free rewrite rules (meaning that the control language itself is specified by a production system) requires $O(2^n)$ productions, thus prooving our theorem.

References

/1/ Georgeff M.P., Procedural Control in Production Systems, Artificial Intelligence (1982) pp. 175-201

/2/ Greibach, S.A., "Theory of Program Structures", Lecture Notes in Computer Science 36, Springer Verlag, Berlin, 1975

/3/ Engelfriet J., Simple Program Schemes and Formal Languages, Lecture Notes in Computer Science 20, Springer Verlag, Berlin, 1974

/3/ Trum P., Wotschke D., "Economy of Description for Program Schemes". InProceedings of FCT 83, Springer Verlag, Berlin, 1983

/4/ Meinecke-Schmidt E., Succinctness of Deeriptions of Context-Free, Regular and Finite Languages, Technical Report DAIMI PB-84, University of Aarhus, 1978