Optimal control of robot behaviour using language measure[1]

Xi Wang, Asok Ray*, Peter Lee and Jinbo Fu

The Pennsylvania State University, University Park,
PA 16802, USA
E-mail: xxw117@psu.edu E-mail: axr2@psu.edu
E-mail: cfl106@psu.edu E-mail: jbfu@psu.edu
*Corresponding author

Abstract: This paper presents optimal control of robot behaviour in the discrete event setting. Real signed measure of the language of supervised robot behaviour serves as the performance index for synthesis of the optimal policy. The computational complexity of control synthesis is polynomial in the number of states of the deterministic finite state automaton model that is generated from the regular language of the unsupervised robot behaviour. The results of simulation experiments on a robotic test bed are presented to demonstrate the efficacy of the proposed optimal control policy.

Keywords: behavioural robotics; discrete event supervisory control; formal languages; learning; performance measure.

Reference to this paper should be made as follows: Wang, X., Ray, A., Lee, P. and Fu, J. (2004) 'Optimal control of robot behaviour using language measure', *Int. J. Vehicle Autonomous Systems*, Vol. 2, Nos. 3/4, pp. 147–167.

Biographical notes: Xi Wang received a PhD degree in mechanical engineering and an MS degree in electrical engineering from The Pennsylvania State University, University Park, PA, in 2003 and 2002, respectively. He received a MS degree in automobile engineering at Tsinghua University, Beijing, China, in 1998 and a BS degree in automobile engineering at Xi'an Jiaotong University, Xi'an, China, in 1993. Dr Wang's research experience and interests include: robotics and manufacturing automation, control and optimisation, mechatronics and instrumentation, intelligent systems and learning, particle filtering, and computer vision. Dr Wang is a member of IEEE.

Asok Ray earned a PhD degree in mechanical engineering from Northeastern University, Boston, MA, and also graduate degrees in each discipline of electrical engineering, mathematics, and computer science. Dr Ray joined the Pennsylvania State University in July 1985, and is currently a Distinguished Professor of mechanical engineering. Dr Ray's research experience and interests include: control and optimisation of continuously varying and discrete-event dynamical systems; intelligent instrumentation for real-time distributed systems; and modelling and analysis of complex dynamical systems from thermodynamic perspectives. Dr Ray has authored or co-authored 370 research publications including over 150 scholarly articles in refereed journals such as *Transactions of ASME*, *IEEE* and *AIAA*, and research monographs. Dr Ray is a Fellow of IEEE, a Fellow of ASME, and an Associate Fellow of AIAA.

Peter Lee earned BSME, MSME and MSEE degrees from Pennsylvania State University, University Park, PA and is currently a Research Assistant in the Networked Robotics Laboratory of the Mechanical Engineering Department at Penn State. Mr. Lee's research experience and interests include: robotics, computers and computer networks.

Jinbo Fu received Bachelor and Master degrees, both in precision instruments and mechanology, from Tsinghua University, Beijing, China, in 1996 and 1999, respectively. He also received Master degrees in electrical engineering and mechanical engineering from Pennsylvania State University, University Park in 2002 and 2003 respectively, and a PhD degree in mechanical engineering from Pennsylvania State University, University Park in 2003. Since 2000, he has been working on the theories and applications of discrete event supervisory (DES) control in several projects. His research interests include DES control theories and application, intelligent systems, engine propulsion system command and control, damage mitigating control, fault detection and information fusion.

1 Introduction

In the discrete event setting, the behaviour of physical plants, such as a robot, is often modelled as regular languages that can be realised by deterministic finite state automata (DFSA) in the discrete event setting (Ramadge and Wonham, 1987). This paper introduces and validates a novel approach for robot behaviour selection based on discrete event supervisory control in terms of the language measure μ (Ray and Phoha, 2003; Surana and Ray, 2004; Wang and Ray, 2004). The controlled sublanguage of a DFSA plant model could be different under different supervisors that are constrained to satisfy different specifications. Such a partially ordered set of sublanguages requires a quantitative measure for total ordering of their respective performance. The language measure serves as a common quantitative tool to compare the performance of different supervisors and is assigned an event cost matrix and a state characteristic vector.

This paper presents discrete event optimal supervisory control of mobile robot operations, where the control policy is synthesised by maximising the language measure of robot behaviour. This novel approach is called μ-optimal in the sequel. In contrast to the Q-learning reinforcement (Watkins, 1989; Watkins and Dayan, 1992) that has been widely used in robotics, computational complexity of μ-optimal control is polynomial in the number of states of the deterministic finite state automaton (DFSA) model of unsupervised robot behaviour. The results of simulation experiments on a robotic test bed are presented for typical scenarios to demonstrate the efficacy of the μ-optimal control policy.

The paper is organised in six sections including the present section. The language measure is briefly reviewed in Section 2 including introduction of the notations. Section 3 presents the parameter identification algorithm and the associated stopping rules. Section 4 proposes a discrete event supervisory (DES) control synthesis strategy. Section 5 validates the synthesis algorithm through simulation on a mobile robotic system. The paper is summarised and concluded in Section 6.

2 Quantitative measure of regular languages for supervisory control

This section introduces the notion of language measure (Ray and Phoha, 2003; Surana and Ray, 2004; Wang and Ray, 2004) that assigns an event cost matrix, denoted as the $\tilde{\mathbf{\Pi}}$-matrix, and a state characteristic vector, denoted as the $\mathbf{X}$-vector. Event costs (i.e. elements of the $\tilde{\mathbf{\Pi}}$-matrix) are based on plant states, where they are generated, are physical phenomena dependent on the plant behaviour, and are conceptually similar to the conditional probabilities of the respective events. On the other hand, the characteristic vector, denoted as the $\mathbf{X}$-vector, is chosen based on the designer's perception of the individual state's impact on the system performance. In the performance evaluation of both the unsupervised and supervised plant behaviour, the critical parameter is the event cost $\tilde{\mathbf{\Pi}}$-matrix. Since the plant behaviour is often slowly time-varying, there is a need for online parameter identification to generate up-to-date values of the $\tilde{\mathbf{\Pi}}$-matrix within allowable bounds of errors.

Let $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$ be a trim (i.e. accessible and co-accessible) finite-state automaton model that represents the discrete event dynamics of a physical plant where $Q = \{q_k : k \in \mathcal{I}_Q\}$, where $\mathcal{I}_Q \equiv \{1, 2, \ldots, n\}$, is the set of states with q_i, where $i \in \mathcal{I}_Q$, being the initial state; $\Sigma = \{\sigma_k : k \in \mathcal{I}_\Sigma\}$, where $\mathcal{I}_\Sigma \equiv \{1, 2, \ldots, m\}$, is the alphabet of events; $\delta : Q \times \Sigma \to Q$ is the (possibly partial) function of state transitions; and $Q_m \equiv \{q_{m_1}, q_{m_2}, \ldots q_{m_l}\} \subseteq Q$ is the set of marked (i.e. accepted) states $q_{m_k} = q_j$ for some $j \in \mathcal{I}_Q$.

Let Σ^* be the Kleene closure of Σ, i.e. the set of all finite-length strings made of the events belonging to Σ as well as the empty string ϵ that is viewed as the identity of the monoid Σ^* under the operation of string concatenation, i.e. $\epsilon s = s = s\epsilon$. The extension $\delta^* : Q \times \Sigma^* \to Q$ is defined recursively in the usual sense (Ramadge and Wonham, 1987).

Definition 2.1

The language $L(G_i)$ generated by a DFSA G initialised at the state $q_i \in Q$ is defined as:

$$L(G_i) = \{s \in \Sigma^* | \delta^*(q_i, s) \in Q\}. \tag{1}$$

The language $L_m(G_i)$ marked by the DFSA G initialised at the state $q_i \in Q$ is defined as:

$$L_m(G_i) = \{s \in \Sigma^* | \delta^*(q_i, s) \in Q_m\}. \tag{2}$$

Definition 2.2

For every $q_j \subset Q$, let $L(q_i,\ q_j)$ denote the set of all strings that, starting from the state q_i, terminate at the state q_j, i.e.

$$L(q_i, q_j) = \{s \in \Sigma^* | \delta^*(q_i, s) = q_j \in Q\}. \tag{3}$$

The set Q_m of marked states is partitioned into Q_m^+ and Q_m^-, i.e. $Q_m = Q_m^+ \cup Q_m^-$ and $Q_m^+ \cap Q_m^- = \emptyset$, where Q_m^+ contains all *good* marked states that we desire to reach, and Q_m^- contains all *bad* marked states that we want to avoid, although it may not always be possible to completely avoid the *bad* states while attempting to reach the *good* states. To characterise this, each marked state is assigned a real value based on the designer's perception of its impact on the system performance.

Definition 2.3

The characteristic function $\chi : Q \to [-1, 1]$ that assigns a signed real weight to state-based sublanguages $L(q_i, q)$ is defined as

$$\forall q \in Q, \quad \chi(q) \in \begin{cases} [-1, 0) & q \in Q_m^- \\ \{0\} & q \notin Q_m \\ (0, 1] & q \in Q_m^+ \end{cases}. \tag{4}$$

The state weighing vector, denoted by $\mathbf{X} = [\chi_1 \chi_2 \cdots \chi_n]^T$, where $\chi_j \equiv \chi(q_j) \forall k$, is called the $\mathbf{X}$-vector. The j-th element χ_j of $\mathbf{X}$-vector is the weight assigned to the corresponding terminal state q_j.

In general, the marked language $L_m(G_i)$ consists of both good and bad event strings that, starting from the initial state q_i, lead to Q_m^+ and Q_m^- respectively. Any event string belonging to the language $L^0 = l(G_i) - L_m(G_i)$ leads to one of the non-marked states belonging to $Q - Q_m$ and L^0 does not contain any one of the good or bad strings. Based on the equivalence classes defined in the Myhill–Nerode Theorem, the regular languages $L(G_i)$ and $L_m(G_i)$ can be expressed as:

$$L(G_i) = \bigcup_{q_k \in Q_m} L(q_i, q_k) = \bigcup_{k=1}^{n} L(q_i, q_k) \tag{5}$$

$$L_m(G_i) = \bigcup_{q_k \in Q_m} L(q_i, q_k) = L_m^+ \cup L_m^- \tag{6}$$

where the sublanguage $L(q_i, q_k) \subset G_i$ having the initial state q_i is uniquely labelled by the terminal state $q_k, k \in \mathcal{I}$ and $L(q_i, q_j) \cap L(q_i, q_k) = \emptyset \forall j \neq k$; and $L_m^+ \equiv \bigcup_{q \in Q_m^+} L(q_i, q)$ and $L_m^- \equiv \bigcup_{q \in Q_m^-} L(q_i, q)$ are good and bad sublanguages of $L_m(G_i)$, respectively. Then, $L^0 = \bigcup_{q \notin Q_m} L(q_i, q)$ and $L(G_i) = L^0 \cup L_m^+ | \cup L_m^-$.

Now we construct a signed real measure $\mu : 2^{L(G_i)} \to R \equiv (-\infty, +\infty)$ on the σ-algebra $K = 2^{L(G_i)}$. Interested readers are referred to Wang and Ray (2004) for the details of measure–theoretic definitions and results. With the choice of σ-algebra, every singleton set made of an event string $\omega \in L(G_i)$ is a measurable set, which qualifies itself to have a numerical quantity based on the above state-based decomposition of $L(G_i)$ into L^0(null), L^+(positive), and L^-(negative) sublanguages.

Conceptually similar to the conditional transition probability, each event is assigned a state-dependent cost.

Definition 2.4

The event cost of the DFSA G_i is defined as a (possibly partial) function $\tilde{\pi} : Q \times \Sigma^* \to [0, 1]$ such that $\forall q_i \in Q$, $\forall \sigma_j \in \Sigma$, $\forall s \in \Sigma^*$,

1 $\tilde{\pi}[q_i, \sigma_j] \equiv \tilde{\pi}_{ij} \in [0, 1)$; $\sum_j \tilde{\pi}_{ij} < 1$;

2 $\tilde{\pi}[q_i, \sigma_j] = 0$ if $\delta(q_i, \sigma_j)$ is undefined; $\tilde{\pi}[q_i, \epsilon] = 1$;

3 $\tilde{\pi}[q_i, \sigma_j s] = \tilde{\pi}[q_i, \sigma_j]\ \tilde{\pi}[\delta(q_i, \sigma_j), s]$.

Definition 2.5

The state transition cost of the DFSA G_i is defined as a function $\pi : Q \times Q \to [0, 1)$ such that $\forall q_i, q_j \in Q, \pi[q_i, q_j] = \sum_{\sigma \in \Sigma : \delta(q_i, \sigma) = q_j} \tilde{\pi}[q_i, \sigma] \equiv \pi_{ij}$ and $\pi_{ij} = 0$ if $\{\sigma \in \Sigma : \delta(q_i, \sigma) = q_j\} = \emptyset$. The $n \times n$ state transition cost Π-matrix is defined as:

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n1} & \pi_{n2} & \cdots & \pi_{nn} \end{bmatrix}.$$

Definition 2.6

The real signed measure μ of every singleton string set $\Omega = \{\omega\} \in 2^{L(G_i)}$ where $\omega \in L(q_i, q)$ is defined as $\mu(\Omega) \equiv \tilde{\pi}[q_i, \omega]\chi(q)$. It follows that the signed real measure of the sublanguage $L(q_i, q) \subseteq L(G_i)$ is defined as

$$\mu\big(L(q_i, q)\big) = \left(\sum_{\omega \in L(q_i, q)} \tilde{\pi}[q_i, \omega] \right) \chi(q). \tag{7}$$

And the signed real measure of the language of a DFSA G_i initialised at a state $q_i \in Q$, is defined as:

$$\mu_i \equiv \mu\big(L(G_i)\big) = \sum_{q \in Q} \mu(L(q, q_i)). \tag{8}$$

The language measure vector, denoted as $\boldsymbol{\mu} = [\mu_1\, \mu_2\, \ldots\, \mu_n]^T$, is called the $\boldsymbol{\mu}$-vector.

It is shown in Surana and Ray (2004) and Wang and Ray (2004) that the signed real measure μ_i can be written as:

$$\mu_i = \sum_j \pi_{ij}\, \mu_j + \chi_i. \tag{9}$$

In vector form, $\boldsymbol{\mu} = \mathbf{\Pi}\boldsymbol{\mu} + \mathbf{X}$ whose solution is given by

$$\boldsymbol{\mu} = (\mathbf{I} - \mathbf{\Pi})^{-1}\mathbf{X}. \tag{10}$$

The inverse in Equation (10) exists because Π is a contraction operator (Surana and Ray, 2004; Wang and Ray, 2004).

2.1 Probabilistic interpretation

Since the plant model is an inexact representation of the physical plant, there exist unmodelled dynamics to be accounted for. This can manifest itself either as unmodelled events that may occur at each state or as unaccounted states in the model. Let Σ_k^u denote the set of all unmodelled events at state k of the DFSA $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$. Let us create a new unmarked absorbing state q_{n+1}, called the dump state (Ramadge and Wonham, 1987), and extend the transition function δ to $\delta_e : (Q \cup \{q_{n+1}\}) \times (\Sigma \cup (\cup_k \Sigma_k^u)) \to (Q \cup \{q_{n+1}\})$ as follows:

$$\delta_e(q_k,\sigma) = \begin{cases} \delta(q_k,\sigma) & \text{if} \quad q_k \in Q \quad \text{and} \quad \sigma \in \Sigma \\ q_{n+1} & \text{if} \quad q_k \in Q \quad \text{and} \quad \sigma \in \Sigma_k^u \\ q_{n+1} & \text{if} \quad k = n+1 \quad \text{and} \quad \sigma \in \Sigma \cup \Sigma_k^u \end{cases}.$$

Therefore the residue $\theta_k = 1 - \sum_j \tilde{\pi}_{kj}$ denotes the probability of the set of unmodelled events Σ_k^u conditioned on the state q_k. The $\mathbf{\Pi}$-matrix is accordingly augmented to obtain a stochastic matrix $\mathbf{\Pi}^{aug}$ as follows:

$$\mathbf{\Pi}^{aug} = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} & \theta_1 \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2n} & \theta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \pi_{n1} & \pi_{n2} & \cdots & \pi_{nn} & \theta_n \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Since the dump state q_{n+1} is not marked, its characteristic value $\chi(q_{n+1}) = 0$. The characteristic vector then augments to $\mathbf{X}^{aug} \equiv [\mathbf{X}^T\ 0]^T$. With these extensions the language measure vector $\boldsymbol{\mu}^{aug} = [\mu_1\, \mu_2 \ldots \mu_n\, \mu_{n+1}]^T = [\boldsymbol{\mu}^T\, \mu_{n+1}]^T$ of the augmented DFSA $G_i^{aug} \equiv (Q \cup \{q_{n+1}\}, \Sigma \cup (\cup_k \Sigma_k^u), \delta_e, q_i, Q_m)$ can be expressed as:

$$\begin{pmatrix} \boldsymbol{\mu} \\ \mu_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{\Pi}\boldsymbol{\mu} + \mu_{n+1}[\theta_1 \ldots \theta_n]^T \\ \mu_{n+1} \end{pmatrix} + \begin{pmatrix} \mathbf{X} \\ 0 \end{pmatrix}. \tag{11}$$

Since $\chi(q_{n+1}) = 0$ and all transitions from the absorbing state q_{n+1} lead to itself, $\mu_{n+1} = \mu(L_m(G_{n+1})) = 0$. Hence, Equation (11) reduces to that for the original plant DFSA G_i. Thus, the event cost can be interpreted as the conditional probability, where the residue $\theta_k = (1 - \sum_j \tilde{\pi}_{kj}) \in (0,1]$ accounts for the probability of all unmodelled events emanating from the state q_k.

3 Estimation of language measure parameters

This section presents a recursive algorithm for identification of the language measure parameters (i.e. elements of the event cost matrix $\tilde{\mathbf{\Pi}}$) (see Definition 2.4) which, in turn, allows computation of the state transition cost matrix $\mathbf{\Pi}$ (see Definition 2.5) and the language measure $\boldsymbol{\mu}$-vector (see Definition 2.6). It is assumed that the underlying physical process evolves at two different time scales. In the fast-time scale, i.e. over a short time period, the system is assumed to be an ergodic, discrete Markov process. In the slowly-varying time scale, i.e. over a long period, the system (possibly) behaves as a non-stationary stochastic process. For such a slowly-varying non-stationary process, it might be necessary to redesign the supervisory control policy in real time. In that case, the $\tilde{\mathbf{\Pi}}$-matrix parameters should be periodically updated.

3.1 A recursive parameter estimation scheme

Let p_{ij} be the transition probability of the event σ_j at the state q_i, i.e.

$$p_{ij} = \begin{cases} P[\sigma_j | q_i], & \text{if } \exists\, q \in Q,\ s.t.\ q = \delta(q_i,\sigma_j) \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

and its estimate be denoted by the parameter $\hat{p}_{ij}$ that is to be identified from the ensemble of simulation and/or experimental data.

Let a strictly increasing sequence of time epochs of consecutive event occurrence be denoted as:

$$\mathcal{T} \equiv \{t_k : k \in \mathbf{N}_0\} \tag{13}$$

where $\mathbf{N}_0$ is the set of non-negative integers. Let the indicator $\psi : \mathbf{N}_0 \times \mathcal{I}_Q \times \mathcal{I}_\Sigma \to \{0, 1\}$ represent the incident of occurrence of an event. For example, if the DFSA was in state q_i at time epoch t_{k-1}, then

$$\psi_{ij}(k) = \begin{cases} 1, & \text{if } \sigma_j \text{ occurs at time epoch } t_k \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

Consequently, the number of occurrences of any event in the alphabet Σ is represented by $\Psi : \mathbf{N}_0 \times \mathcal{I}_Q \to \{0, 1\}$. For example, if the DFSA was in state q_i at the time epoch t_{k-1}, then

$$\Psi_i(k) = \sum_{j \in \mathcal{I}_\Sigma} \psi_{ij}(k). \tag{15}$$

Let $n : \mathbf{N}_0 \times \mathcal{I}_Q \times \mathcal{I}_\Sigma \to \mathbf{N}_0$ represent the cumulative number of occurrences of an event at a state up to a given time epoch. That is, $n_{ij}(k)$ denotes the number of occurrences of the event σ_j at the state q_i up to the time epoch $t_k \in \mathcal{T}$. Similarly, let $N : \mathbf{N}_0 \times \mathcal{I}_Q \to \mathbf{N}_0$ represent the cumulative number of occurrences of any event in the alphabet Σ at a state up to a given time epoch. Consequently,

$$N_i(k) = \sum_{j \in \mathcal{I}_\Sigma} n_{ij}(k). \tag{16}$$

A frequency estimator, $\hat{p}_{ij}(k)$, for probability $p_{ij}(k)$ of the event σ_j occurring at the state q_i at the time epoch t_k, is obtained as:

$$\begin{aligned} \hat{p}_{ij}(k) &= \frac{n_{ij}(k)}{N_i(k)} \\ \lim_{k \to \infty} \hat{p}_{ij}(k) &= p_{ij} \end{aligned}. \tag{17}$$

A recursive algorithm of learning p_{ij} is formulated as a stochastic approximation scheme, starting at the time epoch t_0 with the initial conditions: $\hat{p}_{ij}(0) = 0$ and $n_{ij}(0) = 0$ for all $i \in \mathcal{I}_Q, j \in \mathcal{I}_\Sigma$; and $\Psi_i(0) = 0$ for all $i \in \mathcal{I}_Q$. Starting at $k = 0$, the recursive algorithm runs for $\{t_k : k \geq 1\}$. For example, upon occurrence of an event σ_j at a state q_i, the algorithm is recursively incremented as:

$$\begin{aligned} n_{ij}(k) &= n_{ij}(k-1) + \psi_{ij}(k) \\ N_i(k) &= N_i(k-1) + \Psi_i(k) \end{aligned}. \tag{18}$$

Next it is demonstrated how the estimates of the language parameters (i.e. the elements of event cost matrix $\tilde{\Pi}$) are determined from the probability estimates. As stated earlier in Section 2.1, the set of unmodelled events at state q_i, denoted by $\Sigma_i^u \, \forall i \in \mathcal{I}_Q$, accounts for the row-sum inequality: $\sum_j \tilde{\pi}_{ij} < 1$ (see Definition 2.4).

Then, $P[\Sigma_i^u] = \theta_i \in (0,1)$ and $\sum_i \tilde{\pi}_{ij} = 1 - \theta_i$. An estimate of the (*i*, *j*)th element of the $\tilde{\boldsymbol{\Pi}}$-matrix, denoted by $\hat{\tilde{\pi}}_{ij}$, is approximated as:

$$\hat{\tilde{\pi}}_{ij}(k) = \hat{p}_{ij}(k)(1 - \theta_i)\ \forall j \in \mathcal{I}_\Sigma. \tag{19}$$

Additional experiments on a more detailed automaton model would be necessary to identify the parameters $\theta_i \forall i \in \mathcal{I}_Q$. Given that $\theta_i \ll 1$, the problem of conducting additional experimentation can be circumvented by the following approximation:

A single parameter $\theta \approx \theta_i \forall i \in \mathcal{I}_Q,\ i \in \mathcal{I}_Q$, such that $0 < \theta \ll 1$, could be selected for convenience of implementation. From the numerical perspective, this option is meaningful because it sets an upper bound on the language measure based on the fact that the sup-norm $\| \mu \|_\infty \leq \theta^{-1}$. Note that each row sum in the $\tilde{\boldsymbol{\Pi}}$-matrix being strictly less than 1, i.e. $\sum_j \tilde{\pi}_{ij} < 1$, is a sufficient condition for finiteness of the language measure.

Theoretically, $\tilde{\pi}_{ij}$ is the asymptotic value of the estimated probabilities $\hat{\tilde{\pi}}_{ij}(k)$ as if the event σ_j occurs infinitely many times at the state q_i. However, dealing with finite amount of data, the objective is to obtain a *good* estimate $\hat{p}_{ij}$ of p_{ij} from independent Bernoulli trials of generating events. Critical issues in dealing with finite amount of data are:

- how much data are needed
- when to stop if adequate data are available.

Section 3.2 addresses these issues.

3.2 A stopping rule for recursive learning

A stopping rule is proposed to find a lower bound on the number of experiments to be conducted for the $\tilde{\boldsymbol{\Pi}}$-matrix parameter identification. This section presents such a stopping rule for an inference approximation having a specified absolute error bound ε with a probability λ. The objective is to achieve a trade-off between the number of experimental observations and the estimation accuracy. A robust stopping rule is presented below.

A bound on the required number of samples is estimated using the Gaussian structure for binomial distribution that is an approximation of the sum of a large number of independent and identically distributed (i.i.d.) Bernoulli trials of $\hat{\tilde{\pi}}_{ij}(l)$. The central limit theorem yields $\hat{\tilde{\pi}}_{ij} \sim \mathcal{N}\left(\tilde{\pi}_{ij}, \frac{\tilde{\pi}_{ij}(1-\tilde{\pi}_{ij})}{N}\right)$, where $\mathcal{N}$ indicates normal (or Gaussian) distribution with $E[\hat{\tilde{\pi}}_{ij}] \approx \tilde{\pi}_{ij}$ and $\mathrm{Var}[\hat{\tilde{\pi}}_{ij}] \equiv \sigma^2 \approx \frac{\tilde{\pi}_{ij}(1-\tilde{\pi}_{ij})}{N}$, provided that the number of samples N is sufficiently large. Let $\Delta \equiv \hat{\tilde{\pi}}_{ij} - \tilde{\pi}_{ij}$, then $\frac{\Delta}{\sigma} \sim \mathcal{N}(0,1)$. Given $0 < \varepsilon \ll 1$ and $0 <\ll 1$, the problem is to find a bound N_b on the number N of experiments such that $P\{|\Delta| \geq \varepsilon\} \leq \lambda$. Equivalently,

$$P\left\{\frac{|\Delta|}{\sigma} \geq \frac{\varepsilon}{\sigma}\right\} \leq \lambda \tag{20}$$

that yields a bound N_b on N as:

$$N_b \geq \left(\frac{\xi^{-1}(\lambda)}{\varepsilon} \right)^2 \tilde{\pi}_{ij}(1 - \tilde{\pi}_{ij}) \tag{21}$$

where $\xi(x) \equiv 1 - \sqrt{\frac{2}{\pi}} \int_0^x e^{-\frac{t^2}{2}} dt$. Since the parameter $\tilde{\pi}_{ij}$ is unknown, one may use the fact that $\tilde{\pi}_{ij}(1 - \tilde{\pi}_{ij}) \leq 0.25$ for every $\tilde{\pi}_{ij} \in [0, 1]$ to (conservatively) obtain a bound on N only in terms of the specified parameters ε and λ as:

$$N_b \geq \left(\frac{\xi^{-1}(\lambda)}{2\varepsilon} \right)^2. \tag{22}$$

The above estimate of the bound on the required number of samples is less conservative than that obtained from the Chernoff bound and is significantly less conservative than that obtained from Chebyshev bound (Pradham and Dagum, 1996) that does not require the assumption of any specific distribution of Δ except for finiteness of the rth ($r = 2$) moment.

4 Discrete event supervisory control synthesis in μ measure

In the conventional discrete event supervisory (DES) control synthesis (Ramadge and Wonham, 1987), the qualitative measure of maximum permissiveness plays an important role. For example, under full state observation, if a specification language K is not controllable with respect to the plant automaton G and the set Σ_u of uncontrollable events, then a supremal controllable sublanguage $SupC(K) \subseteq K$ yields maximal permissiveness. However, increased permissiveness of the controlled language $L(S/G)$ may not generate better plant performance from the perspectives of mission accomplishment. This section relies on the language measure μ to quantitatively synthesise a DES control policy. The objective is to design a supervisor such that the controlled plant automaton S/G maximises the performance that is chosen as the measure μ of the controlled plant language $L(S/G)$. The pertinent assumptions for the DES control synthesis are delineated below.

A1: (Cost redistribution) The probabilities of occurrence of *controllable* events in a controlled sublanguage $L(S/G) \subseteq L(G)$ are proportional to those in $L(G)$. For all $q \in Q_S$, where Q_S is the state space of the supervisor automaton S, and $\sigma \in \Sigma_S(q)$, where $\Sigma_S(q)$ is the set of events defined at $q \in Q_S$.

$$\tilde{\pi}_S[q, \sigma] = \frac{\tilde{\pi}_G[q, \sigma]}{\sum_{\sigma \in \Sigma_S(q)} \tilde{\pi}_G[q, \sigma]}. \tag{23}$$

A2: (Event controllability) Any transition $\delta(q, \sigma)$, defined in the plant automaton G such that $\sigma \in \Sigma_{uc}(G)$ and $q \in Q$, is kept enabled in a supervisor S.

Under assumption A1, the sum of event costs defined at the state q of a supervisor S is equal to that of state q of the plant G, i.e.

$$\sum_{\sigma \in \Sigma_S(q)} \tilde{\pi}_S[q, \sigma] = \sum_{\sigma \in \Sigma_G(q)} \tilde{\pi}_G[q, \sigma].$$

Lemma 4.1 (Finiteness)

By disabling controllable events in a plant automaton G, there is only a finite number of controllers $S_i, i \in \mathcal{I}_c$, where $\mathcal{I}_c$ is the set of controllers with cardinality $|\mathcal{I}_c| = n_c$, such that for every $i \in \mathcal{I}_c,\ L(S_i) = L(S_i/G) \subseteq L(G)$.

Proof

Under assumption A2, it suffices to show that the number of all possible permutations of disabling controllable events defined on all states in G is finite. The worst case is that:

- for every state $q \in G$ and every controllable event $\sigma \in \Sigma_c$, the transition $\delta(q,\ \sigma)$ is defined
- every state q in G does not depend on any other state to be accessible from initial state q_0. Then, the number of all possible transitions n_t is given by:

  $$n_t \leq |Q| \times |\Sigma_c| = nm \tag{25}$$

  where $|Q| = n$ is the number of states and $|\Sigma_c| = m$ is the number of controllable events in G. And the number of all possible supervisors is given by

  $$n_c \leq \binom{n_t}{0} + \binom{n_t}{1} + \binom{n_t}{2} + \cdots + \binom{n_t}{n_t} = \sum_{i=0}^{n_t} \binom{n_t}{i} < \infty. \tag{26}$$

Lemma 4.1 shows that there are finitely many supervisors whose generating language is a subset of $L(G)$ given the fact that both the state space and event alphabet are finite. Next we present pertinent results in the form of two theorems. Theorem 4.1 states that out of all possible supervisors constructed from G, there exists a supervisor that maximises the language measure μ with respect to $(\mathbf{\Pi},\ \mathbf{X})$. Theorem 4.2 describes a general transition structure of the μ-optimal supervisor S^*. In particular, at every state in $q \in Q_c(S^*)$ in S^*, there is one and only one controllable event defined.

Theorem 4.1 (Existence) (Fu et al., 2004)

Given a DFSA plant $G = (Q,\ \Sigma,\ \delta,\ q_0,\ Q_m)$, $\mathbf{\Pi}$-matrix, and $\mathbf{X}$-vector, there exist an optimal supervisor S^* such that $\mu(L(S^*/G)) = \max_{i \in \mathcal{I}_c} \mu(L(S_i/G))$.

Theorem 4.2

Given a DFSA plant $G = (Q,\ \Sigma,\ \delta,\ q_0,\ Q_m)$, $\mathbf{\Pi}$-matrix, and $\mathbf{X}$-vector, the event set $\Sigma_c(G,\ q)$ and the plant set $Q_c(G)$ are defined as follows:

$$\Sigma_c(G, q) = \{\sigma \in \Sigma_c \,|\, \delta(q, \sigma) \text{ is defined in } G\} \tag{27}$$

$$Q_c(G) = \{q \in Q \,|\, \Sigma_c(G, q) \neq \emptyset\}. \tag{28}$$

For every state $q \in Q_c(G)$, there is one and only one controllable event left enabled in the μ-optimal supervisor S^*, i.e.

$$\forall q \in Q_c(S^*), \quad |\Sigma_c(S^*, q)| = 1. \tag{29}$$

Proof

Let us assume that there exists a supervisor S^l such that $\mu(L(S^l/G)) > \mu(L(S^*/G))$ and $|\Sigma_c(S^l, q)| > 1$ for some $q \in Q_c(S^l)$.

$$\begin{aligned}
\Delta\boldsymbol{\mu} &\triangleq \boldsymbol{\mu}^i - \boldsymbol{\mu}^* \\
&= \boldsymbol{\mu}(L(S^l/G)) - \boldsymbol{\mu}(L(S^*/G)) \\
&= [\mathbf{I} - \boldsymbol{\Pi}(S^l)]^{-1}\mathbf{X} - [\mathbf{I} - \boldsymbol{\Pi}(S^*)]^{-1}\mathbf{X} \\
&= [\mathbf{I} - \boldsymbol{\Pi}(S^l)]^{-1}\{[\mathbf{I} - \boldsymbol{\Pi}(S^*)] - [\mathbf{I} - \boldsymbol{\Pi}(S^l)]\} \\
&\quad [\mathbf{I} - \boldsymbol{\Pi}(S)^*]^{-1}\mathbf{X} \\
&= [\mathbf{I} - \boldsymbol{\Pi}(S^l)]^{-1}(\boldsymbol{\Pi}(S^l) - \boldsymbol{\Pi}(S^*))\boldsymbol{\mu}^*
\end{aligned}$$

1 $[\mathbf{I} - \boldsymbol{\Pi}(S^l)]^{-1} \geq 0$. By Taylor series expansion,

$$[\mathbf{I} - \boldsymbol{\Pi}(S^l)]^{-1} = \sum_{n=0}^{\infty} (\boldsymbol{\Pi}(S^l))^n. \tag{30}$$

Since each element of $\boldsymbol{\Pi}(S^l)$ is non-negative, so is each element of $(\boldsymbol{\Pi}(S^l))^n$. Therefore, $[\mathbf{I} - \boldsymbol{\Pi}(S^l)]^{-1} \geq 0$ elementwise.

2 Let us suppose $\Sigma_c(S^l, q_j) = \{\sigma_m, \sigma_n\}$ and $\Sigma_c(S^*, q_j) = \{\sigma_l\}$, for some $j \in \mathcal{I}$ then the j-th element of $(\boldsymbol{\Pi}(S^l) - \boldsymbol{\Pi}(S^*))\boldsymbol{\mu}^*$ is given by

$$\begin{aligned}
\Delta_j &= \big([0 \ldots \tilde{\pi}_{jm} \ldots \tilde{\pi}_{jn} \ldots 0] - [0 \ldots \tilde{\pi}_{jl} \ldots 0]\big) \\
&\quad [\ldots \mu^*_m \ldots \mu^*_l \ldots \mu^*_n \ldots]^T \\
&= \tilde{\pi}_{jm}\,\mu_m + \tilde{\pi}_{jn}\,\mu_n - \tilde{\pi}_{jl}\,\mu_l \\
&\leq (\tilde{\pi}_{jm} + \tilde{\pi}_{jn})\mu_\kappa - \tilde{\pi}_{jl}\,\mu_l && (\mu_\kappa = \max\{\mu_m, \mu_n\}) \\
&< (\tilde{\pi}_{jm} + \tilde{\pi}_{jn} - \tilde{\pi}_{jl})\mu_l && (\because\ \mu_m < \mu_l) \\
&< 0 && (\because\ \tilde{\pi}_{jm} + \tilde{\pi}_{jn} = \tilde{\pi}_{jl}).
\end{aligned}$$

Since $\Delta_j < 0$ for every $j \in \mathcal{I}$, $\Delta\boldsymbol{\mu} < 0$. This contradicts the original hypothesis that the supervisor S^* is optimal, i.e. $\mu(L(S/G)) \leq \mu(L(S^*/G))$ for all supervisors S.

Intuitively, at a given state, a control synthesis algorithm should attempt to enable only the controllable event that leads to the next state with highest performance measure μ, equivalently, disabling the rest of controllable events defined at that state, if any. A recursive synthesis algorithm is first presented and then it is shown that $\boldsymbol{\mu}$ is monotonically increasing elementwise on every iteration.

1 Initialisation. Set $\mathbf{\Pi}^0 = \mathbf{\Pi}_p$, then compute $\boldsymbol{\mu}^0 = (\mathbf{I} - \mathbf{\Pi}^0)^{-1}\mathbf{X}$.

2 Recursion. At k-th iteration, where $k \geq 1$,

a μ maximisation. For every $q \in Q_c(G)$, identify the event $\sigma^* \in \Sigma_c(G, q)$ such that $q^* = \delta(q, \sigma^*)$ and

$$\mu\Big(L(S^k(\tilde{\mathbf{\Pi}}^k), q^*)\Big) = \max_{\substack{\sigma\in\Sigma_c(G,q)\\ q'=\delta(q,\sigma)}} \mu\Big(L(S^k(\tilde{\mathbf{\Pi}}^k), q')\Big) \tag{31}$$

where $S^k(\tilde{\mathbf{\Pi}}^k)$ is the intermediate supervisor at k-th iteration whose transition is determined by $\tilde{\mathbf{\Pi}}^k$. Let $\boldsymbol{\sigma}^* = [\sigma_1^* \, \sigma_2^* \ldots \sigma_n^*]^T$, where the i-th element in $\boldsymbol{\sigma}^*$ is the controllable event left-enabled at state q_i of the plant G according to Equation (31).

3 Event Disabling. Disable the event set $\Sigma_c(G, q) - \{\boldsymbol{\sigma}^*(q)\}$ for every $q \in Q_c(G)$ and redistribute event cost according to Equation (23). This results in a new $\tilde{\mathbf{\Pi}}^{k+1}$-matrix which consequently produces a new $\mathbf{\Pi}^{k+1}$-matrix.

$$\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k + \Delta^k \tag{32}$$

where Δ^k records the difference between $\mathbf{\Pi}^{k+1}$ and $\mathbf{\Pi}^{k+1}$, consisting positive and negative event costs corresponding to those kept and disabled controllable events, respectively. The resulting supervisor $S^{k+1}(\tilde{\mathbf{\Pi}}^{k+1})$.

4 Measure Computation. Compute $\boldsymbol{\mu}^{k+1} = (\mathbf{I} - \mathbf{\Pi}^{k+1})^{-1}\mathbf{X}$

5 Termination. If $\tilde{\mathbf{\Pi}}^{k+1} = \tilde{\mathbf{\Pi}}^k$, then stop.

At each iteration of the recursive algorithm, neither the number of states is increased, nor any additional transition is added in the supervisor $S^k(\tilde{\mathbf{\Pi}}^k)$ with respect to the plant automaton G. Therefore, $L(S^k(\tilde{\Pi}^k)) \subseteq L(G)$.

Theorem 4.3 (Monotonicity)

The sequence $\boldsymbol{\mu}^k$, $k = 1, 2, \ldots,$ generated recursively by the above algorithm is monotonically increasing.

Proof

To show that $\{\boldsymbol{\mu}^k, k \in \mathbb{N}\}$ is a monotonic sequence, it suffices to show that $\Delta\boldsymbol{\mu}^k = \boldsymbol{\mu}^{k+1} - \boldsymbol{\mu}^k \geq 0$ for every $k \in \mathbb{N}$.

$$\begin{aligned}
\Delta\boldsymbol{\mu}^k &= [\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1}\mathbf{X} - [\mathbf{I} - \mathbf{\Pi}^k]^{-1}\mathbf{X} \\
&= [\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1}\big[[\mathbf{I} - \mathbf{\Pi}^k] - \mathbf{I} - \mathbf{\Pi}^{k+1}]\big] \\
&\quad\ [\mathbf{I} - \mathbf{\Pi}^k]^{-1}\mathbf{X} \\
&= [\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1}(\mathbf{\Pi}^{k+1} - \mathbf{\Pi}^k)\boldsymbol{\mu}^k \\
&= [\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1}\Delta^k\boldsymbol{\mu}^k.
\end{aligned}$$

1 $[\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1} \geq 0$. By Taylor series expansion,

$$[\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1} = \sum_{n=0}^{\infty} (\mathbf{\Pi}^{k+1})^n. \tag{33}$$

Since each element of $\mathbf{\Pi}^{k+1}$ is non-negative, so is each element of $(\mathbf{\Pi}^{k+1})^n$. Therefore, $[\mathbf{I} - \mathbf{\Pi}^{k+1}]^{-1} \geq 0$ elementwise.

2 $\Delta^k \boldsymbol{\mu}^{k+1} > 0$. For $k \geq 1$, by the recursive algorithm, for any state $q \in Q_c(G)$, there is one and only one left enabled in $S^k(\tilde{\mathbf{\Pi}}^k)$. Let σ_m and σ_n be the only controllable event left enabled on some state $q_l \in Q_c(G)$ at the k-th and $k+1$-th iteration, respectively. Then, only the m-th element and the n-th element of the l-th row are non-zero in $\mathbf{\Pi}^k$ and $\mathbf{\Pi}^{k+1}$, respectively. So the l-th element of $\Delta^k \boldsymbol{\mu}^{k+1}$ is given by:

$$\begin{aligned}\Delta^k \boldsymbol{\mu}^{k+1}(l) &= \left(\begin{bmatrix} 0 \\ \vdots \\ \sum_j \tilde{\pi}_{ij}^{k+1} \\ \vdots \\ 0 \\ \vdots \end{bmatrix} - \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ \sum_j \tilde{\pi}_{ij}^{k} \\ \vdots \end{bmatrix} \right)^T \left[\mu_1^k \cdots \mu_n^k \cdots \mu_m^k \cdots\right]^T . \\ &= \sum_j \tilde{\pi}_{ij} \left(\mu_n^k - \mu_m^k\right) > 0 \end{aligned} \tag{24}$$

The last inequality holds because σ_m is disabled and σ_n is enabled in the $(k+1)$-th iteration only if μ_n is greater than μ_m, according to the recursive algorithm.

Given that every iteration in the above recursive synthesis algorithm generates an intermediate supervisor S^k in the form of $\tilde{\mathbf{\Pi}}^k$ at the kth iteration and its measure μ^k is monotonically increasing with k, the algorithm converges in a finite number of iterations to yield the μ-optimal supervisor S^*. It has been shown by Fu et al. (2004) that the complexity of the above optimal algorithm is polynomial in number of the plant automaton states.

5 Validation of the supervisory control algorithm

This section validates the supervisory control algorithm on a robotic test bed; the control system architecture is shown in Figure 1. The test bed makes use of the Player/Stage.[2] Player is a device server that manages the robot's sensors and actuators, whereas Stage can stimulate a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment. In the simulation experiments, a Pioneer 2 AT robot is equipped with sonar, laser range finder, vision camera, gripper, battery. The DES control module (DCM) communicates with the existing continuous varying control module (CVCM) of the robot by sending and receiving a set of discrete events, listed in Table 1. The DCM is designed to be independent of the underlying physical process such that it provides a mechanism to interact with the CVCM. A DES controller is allowed to plug and play in DCM where the DES controller is loaded in a special format. The CVCM consists of three blocks: continuous-to-discrete (C/D), discrete-to-continuous (D/C), and the CVCM. The

CVCM is connected to the Player via a standard TCP socket for sending and receiving formatted messages that encode up-to-date sensor readings and continuously varying commands of reference signals, respectively, at 10 Hz. The control strategy is event-driven, in which the CVCM generates discrete events based on the continuous sensor data received from the Player. The discrete events are transmitted to DCM in a symbolic form. For example, if the robot is in *search* mode and the incoming vision data indicate the presence of a red unit in its view, the CVCM generates a 'find red unit' event. In response, DCM reacts immediately by sending out a discrete event command back to the CVCM according to the currently loaded supervisor; in this case, the commands are either 'ignore unit' or 'proceed to supply'. After receiving a particular event from the DES controller, the CVCM sends out a set of continuous reference signals to Player for completion of this behaviour of accordingly manoeuvring the robot. The robot continues executing this behaviour until the sensor readings trigger the occurrence of a new event.

Figure 1 Pioneer 2 AT DES simulation block diagram

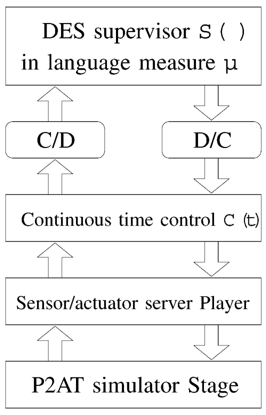


Table 1 List of discrete events

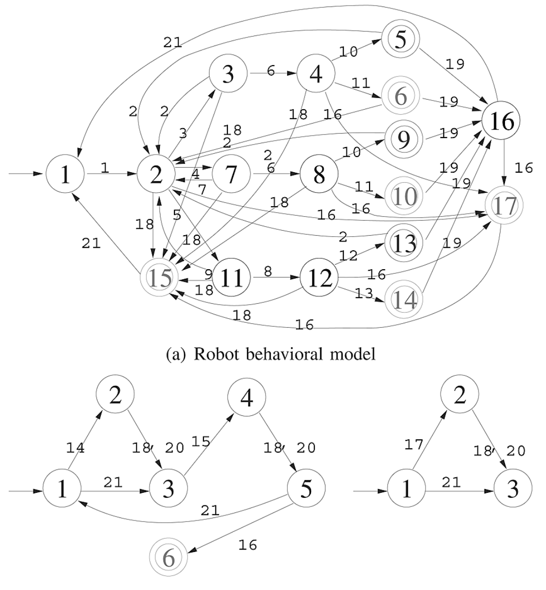
Σ	*Description*	Σ	*Description*
σ_1	start mission	σ_{12}	win the fight
σ_2	search	σ_{13}	loose the fight
σ_3	find blue unit	σ_{14}	battery power medium
σ_4	find pink unit	σ_{15}	battery power low
σ_5	find enemy	σ_{16}	battery power dead
σ_6	proceed to supply	σ^{17}	detected gripper fault
σ_7	ignore unit	σ_{18}	abort mission
σ_8	fight enemy	σ_{19}	return
σ_9	avoid enemy	σ_{20}	ignore anomaly
σ_{10}	finish supply	σ_{21}	return successfully
σ_{11}	fail supply		

The experimental scenario consists of a single robot performing logistic supply and combat operation in a simulated battle field. There are two friendly units (represented by red and green coloured circles) and one enemy (represented by blue circle), which are at stationary locations in the field. The robot does not have prior knowledge of the environment. When a 'start mission' signal is received, the robot randomly searches for a red or green unit. When finding a unit, the robot can either proceed to supply or ignore the unit and keep on searching. It may also encounter an enemy during the course of searching. The robot may decide either to avoid or to fight the enemy. In both cases, there are chances that the robot may fail to supply or lose the fight.

A gripper failure is modelled to signify the robot's failure to complete the task of supplying units. However, fighting with the enemy is still possible. In addition, during the mission, the battery consumption rate changes according to the robot's current actions. For example, the robot's power consumption is higher during the fight with the enemy than supplying the units. The robot needs to return to its base (represented by a large pink circle) to be recharged before the battery voltage drops below a certain level, or the robot is considered to be lost. After each successful return to the base, the robot is reset to normal conditions including full battery charge and normal gripper status. If the robot is incapacitated either due to battery over-drainage or damage in a battle, the 'death toll' is increased by one. In both cases, the mission is automatically restarted with a new robot.

Due to the independence of the robot operation, battery consumption, and occurrence of gripper failures, the entire interaction between robot and environment is modelled by three different submodels, as shown in Figure 2(a) and (b). The blue-coloured and red-coloured states are *good* marked states in Q_m^+ and *bad* marked states in Q_m^-, respectively. Then the models are composed by synchronous composition operator defined below to generate the integrated plant model $G = G_1 \parallel G_2 \parallel G_3$.

Figure 2 Finite state models of robot operation



(a) Robot behavioral model

(b) Battery failure model and gripper failure model

Definition 5.1

Given $G_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1})$, $G_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2})$, synchronous composition of G_1 and G_2, denoted $G_1 \parallel G_2 = (Q, \Sigma, \delta, q_0, Q_m)$, is defined as $Q = Q_1 \times Q_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $q_0 = (q_{0,1}, q_{0,2})$, $Q_m = Q_{m,1} \times Q_{m,2}$ and for every $q = (q_1, q_2) \in Q, \sigma \in \Sigma$

$$\delta(q,\sigma) = \begin{cases} (\delta_1(q_1,\sigma), \delta_2(q_2,\sigma)) & \sigma \in \Sigma_1 \cap \Sigma_2 \\ (\delta_1(q_1,\sigma), q_2) & \sigma \in \Sigma_1 - \Sigma_2 \\ (q_1, \delta_2(q_2,\sigma)) & \sigma \in \Sigma_2 - \Sigma_1 \\ \text{undefined} & \text{otherwise} \end{cases}. \tag{34}$$

After eliminating the inaccessible states, the discrete-event model of the plant automation G consists of 139 states; and there are 21 events as listed in Table 1. The event cost matrix $\tilde{\Pi}$ is then identified by Monte Carlo simulation over 1200 missions according to the parameter identification procedure; convergence of selected non-zero elements in $\tilde{\Pi}$-matrix is demonstrated in Figure 3. For those states that have more than one controllable event defined, the probabilities of occurrence are assumed to be equally distributed. A vast majority of the plant states are unmarked; consequently, the corresponding elements, χ_i, of the characteristic vector are zero. Negative characteristic values, χ_i are assigned to the bad marked states. For example, the states in which the robot is dead due to either losing the battle to the enemy or running out of battery is assigned the negative value of -1. Similarly, positive values are assigned to good states. For example, the state in which the robot wins a battle is assigned 0.5 and the state of successfully providing supplies is assigned 0.3. Using the recursive synthesis algorithm in Section 4, the μ-optimal supervisor S^* is then synthesised. The optimal DES control algorithm converges at the fourth iteration, as listed in Table 2. For the purpose of performance comparison, two additional supervisors, S_1 and S_2 under the following specifications are designed.

Figure 3 Some non-zero elements of $\tilde{\Pi}$-matrix

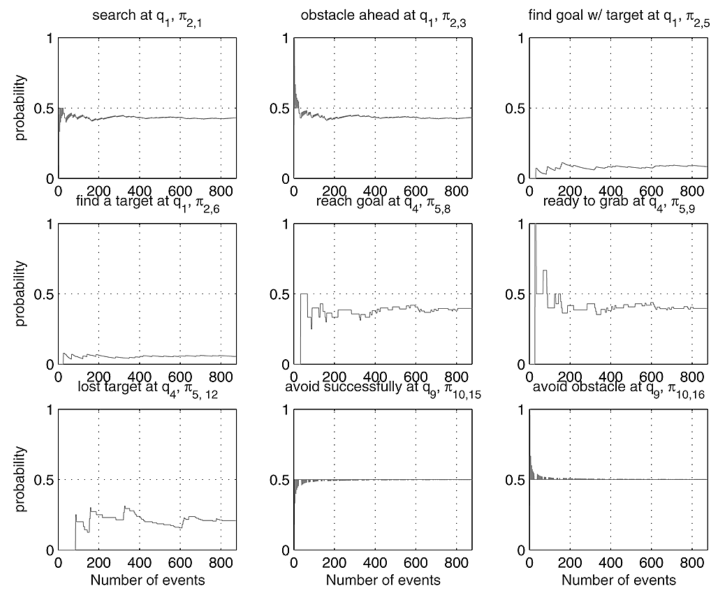


Table 2 Iteration of μ synthesis

k-th iteration	$\boldsymbol{\mu}^k$
0	-1.7834
1	2.8306
2	4.5655
3	4.5696
4	4.5696

The specifications of Controller S_1 are as follows:

- avoid enemy when the battery power is not below medium
- abort all operation when the battery power is low
- if there is a gripper failure, do not supply a discovered unit or abort supply if the supply is ongoing.

The specifications of Controller S_2

- abort all operation if battery power is not below medium
- abort all operation if a gripper failure is detected.

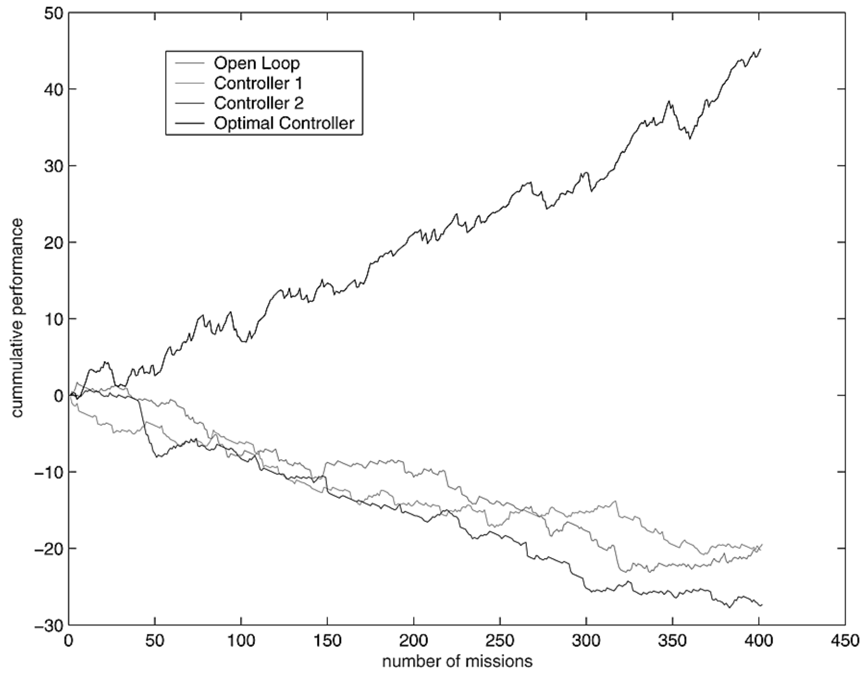
The conventional DES controllers, S_1 and S_2, have 109 and 54 states, respectively, and 400 missions were simulated for the open loop plant and each of the three DES controllers: μ-optimal supervisor S^*, and the conventional S_1 and S_2. The statistics of the simulation results are summarised in Table 3, where the supervised robot yields higher performance under S^* than under S_1 and S_2 or the null supervisor (i.e. the unsupervised plant G). In the last row of Table 3, the cumulative performance of robot operations, with the initial state q_1, is obtained by summing the language measure over 400 missions as: $\sum_{i=1}^{400} \mu_1(i)$.

Table 3 Simulation statistics of 400 missions

Items	G	S_1	S_2	S^*
$\frac{\text{proceed to supply}}{\text{\# of units found}}$	$\frac{150}{434}$	$\frac{134}{408}$	$\frac{137}{367}$	$\frac{482}{556}$
	34.56%	32.84%	37.33%	*86.69%*
finish supply	112	88	97	362
$\frac{\text{win enemy}}{\text{fight enemy}}$	$\frac{37}{65}$	$\frac{33}{70}$	$\frac{22}{68}$	$\frac{69}{167}$
μ	-1.7834	-1.4428	-1.8365	*4.5696*
$\sum_{i=1}^{400} \mu_1(i)$	-20.25	-19.45	-27.3	*45.25*

During these 400 missions, S^* decides to *proceed to supply* for 482 times, three times more than S_1 or S_2. In addition, the probability of deciding to *proceed to supply*, when the robot sees a unit, is much higher than S_1 or S_2. However, the price of this decision is that the robot is likely to drain out the battery energy and therefore may risk being immobile in the middle of the mission. The μ-optimal supervisor S^* also decides to fight many more times (167) than any other supervisor since the reward to win a battle is large ($\chi = 0.5$). Certainly, the number of robots lost under S^* is also higher (48) than that under other supervisors because S^* has to expose the robot more often to the enemy attack to win the battles. On average, S^* outperforms G, S_1 and S_2 in measure $\boldsymbol{\mu} = (\mathbf{I} - \mathbf{\Pi})^{-1}\mathbf{X}$. The cumulative performance of S^* is found to be superior to the cumulative performance of two supervisors and the null supervisor (i.e. open loop or unsupervised robot) over 400 missions as seen in Figure 4. The oscillations in the performance profile, as a function of the number of missions, are attributed to unavailability of the robot resulting from drained battery or damage in the battle.

Figure 4 Cumulative performance comparison of all controllers



6 Related work on *Q*-learning

This section presents the concept of *Q*-learning (Watkins, 1989) that is widely used for reinforcement learning in behaviour-based robotics (Arkin, 1998) for its algorithmic simplicity and ease of transformation from a state function to an optimal control policy. In addition, similar to the approach presented in this section, it does not require a world model. It is well suited for use in stationary, single-agent, fully observable environments that are modelled by Markov Decision Processes (MDPs). However, it often performs well in environments that violate these assumptions. Mahadevan and Connell (1991) have used Q-learning to teach a behaviour-based robot how to push boxes around a room without getting stuck. The mission is heuristically divided into three behaviours: *finding a box*, *pushing a box*, and *recovering from stalled situations*. The results show that the mission decomposition method is capable of learning faster than a method that treats the mission as a single behaviour. Martinson et al. (2002) demonstrated by both simulation and robotic experiments that Q-learning can be used for robot behavioural selection to optimise the overall mission in terms of Q-value. The basic principles of Q-learning are briefly described below.

A *Markov decision process* is a tuple $M = (S, A, P, R)$, where S is the set of states; A is the set of actions; $P : S \times A \times S \to [0, 1]$ the transition probability function; and $R : S \times A \to \mathbb{R}$ is an excepted reward function. Let $\pi : S \to A$ be a mapping between what has happened in the past and what has to be done at the current state. The value of the policy π, starting at state s, is evaluated as:

$$V^{\pi}(s) \equiv E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \tag{35}$$

$$= R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^{\pi}(s') \tag{36}$$

where the subscript or superscript t indicates a time epoch; r is the reward for taking a transition; $\gamma \in [0, 1)$ is the discount factor; $R(s,a)$ is the expected instantaneous reward by action a at state s; and $P(s, a, s')$ is the probability of making a transition from state s to state s' by action a.

The objective is to find an optimal control policy π^* that maximises the future reward with a discount factor γ. One of the important results, used in the Q-learning, is that there exists an optimal stationary policy π^* that maximises $V^{\pi}(s)$ for all states s, where the optimal policy V^{π^*} is denoted by V^*.

Let $Q(s, a)$ denote the discounted reinforcement of taking action a at state s following a policy π. Using the notation of Watkins (1989), $Q^*(s, a)$ is the value of $Q(s, a)$, starting at state s and taking action a and from there on following the optimal policy π^*. Therefore,

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^*(s) \tag{37}$$

$$\pi^* = \arg\max_{a} Q^*(s,a). \tag{38}$$

At each state, the best policy π^* is to take an action with the largest Q-value, i.e. $V^*(s) = \max_a Q^*(s, a)$. The basic idea of Q-learning is to maintain an estimate $\hat{Q}(s, a)$ of $Q^*(s, a)$. The online Q-learning update rule is:

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a)\right) \tag{39}$$

where α is the learning rate. In Watkins (1989) and Watkins and Dayan (1992), it has been shown that the Q values will converge with probability 1 to Q^* if each action is executed in each state an infinite number of times on an infinite run and α is decayed appropriately. Table 4 lists a comparison of the salient features of the Q-learning and language-based optimal methods.

Table 4 Comparison of Q-learning and μ-optimal control methods

Items	*Q-learning*	*μ-optimal*
Model structure	$M = (S, A, T, R)$	$G = (Q, \Sigma, \delta, q_0, Q_m$
Control policy	Markov Decision Process	Discrete Event supervisory
Cost functional	$\arg\max_a Q^*(s, a)$	$\max_{s \in L(G)} \mu(s)$
Transition probability	required $P(s, a, s')$	required $\tilde{\pi}([q, \sigma]$
Reward	on transition $R(s, a)$	on state $\chi(q)$
Discount factor γ	ad hoc	none
Learning rate α	ad hoc	none
Online adaption to dynamic environment	recursive	recursive (ϵ, δ)-threshold redesign
Computational complexity	exponential in S and A	polynomial in Q

7 Summary and conclusions

This paper presents an optimal policy for discrete event supervisory (DES) control of robot behaviour, called μ-optimal, based on a real signed measure μ of regular languages (Ray and Phoha, 2003; Surana and Ray, 2004; Wang and Ray, 2004). The recursive supervisor synthesis algorithm is formulated as an extension of the earlier work of Fu et al. (2004). A robot simulation test bed has been developed for validation of this new concept of DES control policy that maximises the performance index μ and yields better performance than two other supervisors (that are designed in the conventional way; Ramadge and Wonham, 1987), and the null supervisor (i.e. behaviour of the unsupervised robot).

Salient features of μ-optimal control are compared with those of Q-learning reinforcement (Watkins, 1989; Watkins and Dayan, 1992) that has been widely used in robotics. The reward/penalty function $R(s, a)$ in Q-learning is defined for every transition at each state of the Markov decision process based on human perception, largely similar to the characteristic value χ defined for every state of the plant automaton G in the formulation of the language measure μ. That is, in μ-optimal control, a weight is assigned to each state of the automaton similar to what is done in each transition of Q-learning reinforcement. However, while computational complexity of μ-optimal control is polynomial in number of plant automaton states, that of Q-learning reinforcement increases exponentially with the number of internal states and the number of actions (events).

References

Arkin, R.C. (1998) *Behavior Based Robotics*, MIT Press.

Fu, J., Ray, A. and Lagoa, C.M. (2004) 'Unconstrained optimal control of regular languages', *Automatica*, Vol. 40, No. 4, pp.639–648.

Mahadevan, S. and Connell, J. (1991) 'Automatic programming of behavior-based robots using reinforcement learning', *Proceedings of AAAI-91*, pp.768–773.

Martinson, E., Stoytchev, A. and Arkin, R.C. (2002) 'Robot behavioral selection using q-learning', *IEEE International Conference on Robots and Systems* (Lausanne), September 2002.

Pradhan, M. and Dagum, P. (1996) 'Optimal Monte Carlo estimation of belief network inference', *Twelfth Conference on Uncertainty in Artificial Intelligence*, Portland, OR, pp.446–453.

Ramadge, P.J. and Wonham, W.M. (1987) 'Supervisory control of a class of discrete event processes', *SIAM J. Control and Optimization*, Vol. 25, No. 1, pp.206–230.

Ray, A. and Phoha, S. (2003) 'Signed real measure of regular languages for discrete event automata', *Int. J. Control*, Vol. 76, No. 18, pp.1800–1808.

Surana, A. and Ray, A. (2004) 'Signed real measure of regular languages', *Demonstratio Mathematica*, Vol. 37, No. 2, in press.

Wang, X. and Ray, A. (2004) 'A language measure for performance evaluation of discrete event supervisory control systems', *Applied Mathematical Modelling*, in press.

Watkins, C.J.C.H. (1989) 'Learning from delayed rewards', *PhD thesis*, King's College, Cambridge, UK.

Watkins, C.J.C.H. and Dayan, P. (1992) 'Q-learning', *Machine Learning*, Vol. 8, No. 3, pp.279–292.

Notes

[1] This work has been supported in part by Army Research Office (ARO) under Grant No. DAAD19 01 1 0646; and NASA Glenn Research Center under Grant No. NNC04GA49G.

[2] Player/Stage was developed at University of Southern California and is available under GNU Public License at http://playerstage.sourceforge.net/