Uma Metodologia de Projeto de Protocolos de Comunicação Baseada em Técnicas Evolutivas

Sérgio G. Araújo, Aloysio C. P. Pedroza, Antônio C. Mesquita F.

***Resumo*—Este trabalho utiliza Programação Genética para gerar automaticamente máquinas de estados finitos que descrevem a especificação do protocolo a partir do modelo de serviço do protocolo. A metodologia reduz computações aplicadas a autômatos de estados finitos utilizadas nos métodos de síntese tradicionais, abstraindo o projetista de protocolos destas tarefas.**

***Palavras-Chave*—Síntese de Protocolos, Programação Genética, Máquinas de Estados Finitos.**

***Abstract*—This work uses Genetic Programming to automatically generate finite-state models that synthesize protocol specifications from protocol service specification. The approach reduces computation applicable to finite-state automata in traditional protocol synthesis methods, abstracting the protocol designer from such tasks.**

***Index Terms*—Protocol Synthesis, Genetic Programming, Finite-State Machines.**

I. INTRODUÇÃO

Nos *sistemas distribuídos* computadores interconectados trocam informações entre si com o objetivo de prover serviços aos usuários, criando nestes a ilusão de um único servidor centralizado. Os *protocolos de comunicação* são componentes fundamentais destes sistemas. Um protocolo de comunicação é definido como um conjunto de regras que governa o formato e o significado de quadros, mensagens ou pacotes que são trocados através de entidades pares [1].

Os sistemas distribuídos são especificados segundo dois níveis de abstração [2]. No nível de *especificação de serviço*, entidades de protocolo e canais de comunicação conectando tais entidades estão escondidos. A especificação neste nível é descrita exclusivamente através de trocas de *primitivas de serviço* em pontos de acesso ao serviço (SAPs: *Service Access Points*). No nível de *especificação de protocolo*, para cada entidade de protocolo, suas próprias ações, as mensagens trocadas com outras entidades de protocolo e a correta ordenação destas mensagens são descritas. A especificação do serviço do protocolo e a especificação do protocolo devem ser equivalentes, ou seja, elas devem exibir o mesmo comportamento aos usuários do serviço.

Sérgio G. Araújo, Escola de Eng. Elétrica, UFG, Goiânia, E-mail: granato@eee.ufg.br; Aloysio C. P. Pedroza e Antônio C. Mesquita F., Departamento de Eng. Elétrica, UFRJ, Rio de Janeiro, E-mails: aloysio@gta.ufrj.br e mesquita@coe.ufrj.br.

A *síntese de protocolo* é uma importante etapa no ciclo de engenharia de protocolo (Figura 1), podendo ser definida como a geração da especificação do protocolo a partir da especificação do serviço do protocolo.

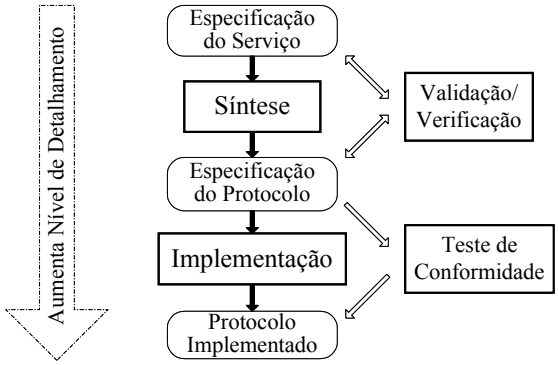

Fig. 1. Ciclo de engenharia de protocolo.

Duas propriedades devem ser garantidas no projeto de protocolos de comunicação [3]: segurança (*safety*) e vivacidade (*liveness*). A primeira diz respeito à correção sintática, garantindo que o protocolo não entrará em estado ou situação indesejados. A propriedade de vivacidade diz respeito à correção semântica, assegurando que o protocolo provê as funções a serem fornecidas aos usuários, especificadas na descrição do serviço do protocolo.

Inúmeras estratégias têm sido propostas com o objetivo de gerar automaticamente especificações de protocolo a partir de especificações de serviço do protocolo, principalmente para modelos de FSM, LOTOS e Redes de Petri, e suas extensões. Como destacado em [4], estas estratégias têm-se preocupado, principalmente, em implementar fluxos de controle complexos [5], suportar restrições de tempo [6] e gerenciar recursos distribuídos [7].

Por outro lado, trabalhos recentes [8, 9] têm obtido sucesso em sintetizar máquinas seqüenciais de pequeno porte com a ajuda de algoritmos genéticos. Em [8] uma abordagem para sintetizar circuitos lógicos seqüenciais a partir de seqüências parciais de entrada/saída é proposta. A referência [9] utilizou representação baseada em estado como alternativa à representação baseada em tecnologia, empregada em [8].

Em [10] gramáticas formais são exploradas como alternativa de evolução (projeto) de autômatos. Nesta abordagem, conhecida como programação genética (PG) orientada à gramática ou G^3P (*Grammar-Guided Genetic Programming*), programas são produzidos combinando gramática livre de contexto (CFG: *Context-Free Grammar*)

com PG. *Genetic Programming Kernel* (GPK) [11] é um sistema G^3P complexo e foi utilizado como núcleo do sistema PG neste trabalho. GPK evolui programas (ou estruturas) em uma linguagem (ou notação) formal desde que uma descrição em BNF seja fornecida na entrada.

As abordagens de síntese de protocolos de comunicação que utilizam o formalismo de máquinas de estados finitos estão completando mais de duas décadas. No entanto, suas execuções implicam em realizar várias transformações em autômatos até chegar à especificação das entidades de protocolos. Este trabalho propõe uma metodologia alternativa para elevar o nível de abstração do projeto de protocolos de comunicação.

A Seção 2 define máquinas de estados finitos e fornece um exemplo de especificação de serviço de protocolo utilizando este formalismo; conceitos básicos de computação evolutiva são introduzidos. A Seção 3 apresenta a metodologia de projeto de protocolos de comunicação. Na Seção 4, um protocolo orientado à conexão é utilizado para exemplificar a metodologia. Na Seção 5 são apresentadas as conclusões.

II. Definições e Conceitos Básicos

A. FSM e Especificação de Serviço do Protocolo

Uma máquina de estados finitos (FSM: *Finite-State Machine*) é definida por uma 7-tupla $\{Q,V,t,q_0,V',o,D\}$ onde $Q \neq \emptyset$ é um conjunto finito de estados, V é um conjunto finito de entradas, t é a função de transição de estados, $q_0 \in Q$ é o estado inicial, V' é um conjunto finito de saídas, o é a função de saída e D é o domínio de especificação, sendo este último um subconjunto de $Q \times V$. t e o caracterizam o comportamento da FSM, ou seja, $t(q,v):Q \times V \rightarrow Q$ e $o(q,v):Q \times V \rightarrow V'$. Se $D = Q \times V$, então t e o são definidos para toda combinação estado atual-entrada possível e, portanto, a FSM é denominada *completamente especificada*.

O serviço a ser fornecido por um protocolo de comunicação é especificado por um conjunto de *primitivas de serviço* disponível ao usuário para acessar o serviço. A Figura 2a fornece o modelo global da *especificação de serviço* (*ES*) de um protocolo orientado à conexão, utilizando uma FSM. As Figuras 2b e 2c fornecem a mesma especificação, segundo a perspectiva da entidade. Nestas figuras, uma primitiva A_i ocorre no *site i*; um envio (para o usuário) e uma recepção (do usuário) da primitiva A é denotado $!A$ e $?A$, respectivamente. A conexão é iniciada pelo emissor (*C_Req*) e o receptor pode aceitá-la (*C_Resp*) ou rejeitá-la (*D_Req*). Caso o serviço esteja estabelecido, o emissor pode iniciar o procedimento de reinicialização (*R_Req*) para remover todas as mensagens do meio de comunicação. Finalmente, a conexão pode ser liberada pelo emissor, ou pelo receptor, através da primitiva de serviço *D_Req*.

B. Computação Evolutiva

A computação evolutiva (CE) consiste de uma classe de algoritmos probabilísticos inspirados em princípios de seleção natural que se beneficiam do aumento do poder de simulação para resolver problemas complexos e aplicações de descobrimento de conhecimento. Exemplos de CE são algoritmos genéticos (AGs) e programação genética (PG).

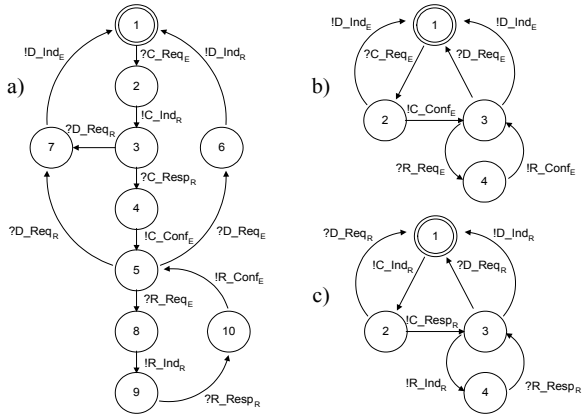

Fig. 2. (a) *ES* global, (b) *ES* do emissor e (c) *ES* do receptor.

O AG emula a maneira com que a natureza apura as características dos seres vivos. O *cromossomo* é o componente básico de um AG. Ele representa um ponto, chamado neste contexto de um *indivíduo*, do espaço de busca (ou solução) do problema. A aptidão é uma medida utilizada para indicar quão próximo um indivíduo está da solução. Pela aplicação iterativa de operadores genéticos (cálculo da aptidão, seleção baseada na aptidão, reprodução, cruzamento e mutação) em uma *população* de indivíduos iniciada randomicamente, tal população evolui no sentido de gerar soluções potenciais para o dado problema.

O AG tem sido aplicado em tarefas de otimização complexas e de muitas variáveis, onde outras estratégias normalmente fracassam. A PG [12] é derivada do AG clássico, sendo que a principal diferença está na representação da solução. Ao contrário do AG, a PG utiliza genótipos de tamanhos variados o que lhe confere maior potencial na criação de estruturas.

As técnicas evolutivas vêm assumindo lugar de destaque entre os sistemas automáticos de aprendizado, beneficiada pelo alto desempenho dos sistemas computacionais. Uma de suas vantagens mais significativas está na possibilidade de resolver problemas pela simples descrição matemática, por meio de uma *função objetivo*, do quê se quer ver presente na solução, não havendo necessidade de indicar explicitamente os passos até o resultado.

III. Metodologia

Em vez de criar uma especificação global de estados do protocolo e projetá-la no alfabeto de eventos de cada *site* (entidade), como em [3, 6], o sistema gera as entidades de protocolo separadamente. A metodologia foca na parte de controle do protocolo, excluindo aspectos de dados, como por exemplo, mensagens com parâmetros. Tal restrição é adotada na maioria das abordagens de síntese de protocolos [13]. As etapas envolvidas na metodologia são mostradas na Figura 3. Inicialmente, uma população de soluções candidatas é criada randomicamente a partir de uma descrição BNF. Em seguida, cada indivíduo da população é avaliado através de uma função objetivo, que lhe atribui uma aptidão. Caso ao menos

um indivíduo apresente a aptidão desejada, indicando que suas saídas são iguais às saídas especificadas quando o autômato é estimulado pela mesma seqüência de entrada, o sistema termina exteriorizando a FSM que, a priori, modela o sistema (Cf. Subseção IV.D). Caso contrário, os operadores genéticos são aplicados (bloco GPK) e uma nova população é gerada para avaliação.

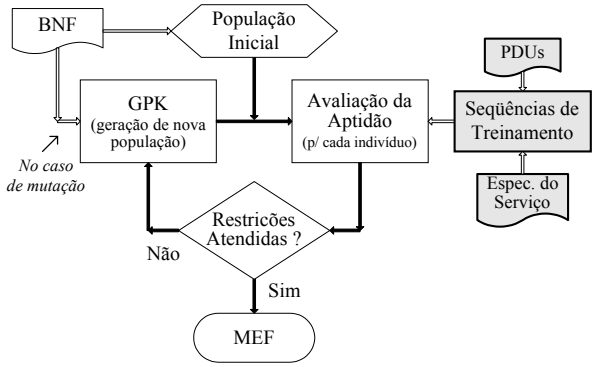

Fig. 3. Metodologia de projeto de protocolos de comunicação

O modelo adotado para o cálculo de aptidão está esboçado na Figura 4. Este modelo é similar à abordagem de "caixa preta" utilizada em testes de conformidade de protocolos de comunicação. Utilizando uma interface estimulada por eventos, a *entidade a ser evoluída* se comunica com o usuário do serviço através de primitivas de serviço e com a entidade par através PDUs. Neste modelo é assumido que o meio de comunicação é confiável, não havendo, portanto, ocorrência de falhas na rede. Tal restrição também é adotada na maioria das abordagens de síntese de protocolos [13]. Em cada geração o sistema estimula uma população de *entidades a serem evoluídas* com seqüências de entradas particulares e armazena as seqüências de saídas correspondentes. Estas seqüências de saídas são comparadas com as saídas corretas e um valor de aptidão é atribuído a cada indivíduo.

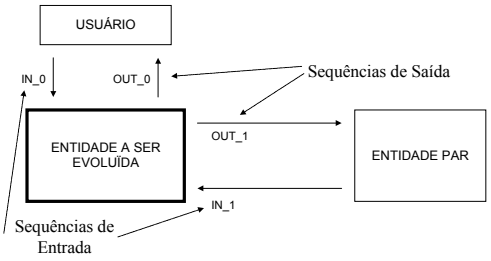

Fig. 4. Modelo para o cálculo de aptidão.

A. Derivação das Seqüências de Treinamento

Uma *seqüência de treinamento ST* é uma seqüência de pares de entrada/saída *corretos* $< v_1/v_1', v_2/v_2', \ldots, v_L/v_L' >$, onde $v \in V$, $v' \in V'$ e L é o comprimento da seqüência. Ao lado de (1) *ES* (modelo global da especificação de serviço), assumido seqüencial, isto é, há uma relação de causalidade entre dois eventos consecutivos, e livre de *deadlock* e *livelock*, os seguintes *elementos de protocolo* devem ser definidos para a derivação completa de uma *ST*:

(2) Restrições do protocolo e
(3) PDUs e respectivas associações às primitivas de serviço.

Um protocolo orientado à conexão é utilizado como exemplo. Nas asserções seguintes, EP_E denota a entidade emissora do protocolo orientado à conexão, ES_E denota o lado emissor da *ES* e $EPRO_E$ denota a especificação da entidade de protocolo para a EP_E. **(1)** *ES* é mostrado na Figura 2a. **(2)** As restrições de protocolo são: **(2.1)** cada estado de uma EP_E deve ser capaz de responder a cada entrada, ou seja, o resultado do processo de síntese é uma FSM completamente especificada, **(2.2)** somente as fases de estabelecimento, reinicialização e liberação de conexão são consideradas **(2.3)** somente a EP_E pode iniciar e reiniciar a conexão. **(3)** As PDUs, juntamente com as primitivas de serviço empregadas na ES_E (Figura 2b), são listadas na Tabela 1, que define os alfabetos de entrada $V=\{0,1,2,3,4,5\}$ e de saída $V'=\{0,1,2,3,4,5,6\}$ da EP_E. **(3.1)** A associação entre primitivas de serviço e PDUs é definida pelo conjunto $\{C\_Req_E\text{-}con\_req,\ C\_Resp_R\text{-}con\_acc,\ D\_Req_E\text{-}dis\_ind,\ D\_Req_R\text{-}dis\_ind,\ R\_Req_E\text{-}rei\_req,\ R\_Resp_R\text{-}rei\_acc\}$.

Tabela 1: PDUs e primitivas de serviço da EP_E.

Entrada (V)	**descrição**	**tipo**	**codificação**
C_Req	Connect_Request	Primitiva	0
D_Req	Disconnect_Request	Primitiva	1
con_acc	connect_accept	PDU	2
dis_ind	disconnect_indication [a]	PDU	3
R_Req	Reinitialization_Request	Primitiva	4
rei_acc	reinitialization_accept	PDU	5
Saída (V')	**descrição**	**tipo**	**codificação**
C_Conf	Connect_Confirm	Primitiva	0
D_Ind	Disconnect_Indication	Primitiva	1
con_req	connect_request	PDU	2
dis_ind	disconnect_indication	PDU	3
R_Conf	Reinitialization_Confirm	Primitiva	4
rei_req	reinitialization_request	PDU	5
Null	"No output"	-	6

Método de Derivação:

Passo_1: Uma seqüência de primitivas de serviço é criada a partir da *ES* (Figura 2a). Neste passo é desejado que todos os caminhos da *ES* estejam presentes. Para este propósito pode ser utilizado um gerador automático de seqüências de teste, programado para gerar seqüências corretas a partir da *ES*.

Exemplo: Percorrendo os estados 1, 2, 3, 4, 5, 7, 1, 2, 3, 7, 1, 2, 3, 4, 5, 8, 9, 10, 5 e 6 da *ES*, a seguinte seqüência de primitivas de serviço é obtida:

<C_Req_E, C_Ind_R, C_Resp_R, C_Conf_E, D_Req_R, D_Ind_E, C_Req_E, C_Ind_R, D_Req_R, D_Ind_E, C_Req_E, C_Ind_R, C_Resp_R, C_Conf_E, R_Req_E, R_Ind_R, R_Resp_R, R_Conf_E, D_Req_E, D_Ind_R>.

Passo_2: A PDU associada é inserida entre todo par de primitivas de serviço consecutivas que ocorrem em *sites* diferentes, exceto quando uma escolha entre primitivas de serviço descendentes é realizada pelo usuário.

Exemplo: A partir da associação primitiva/PDU acima definida, as PDUs são inseridas na seqüência:

<C_Req_E, **con_req**, C_Ind_R, C_Resp_R, **con_acc**, C_Conf_E, D_Req_R, **dis_ind**, D_Ind_E, C_Req_E, **con_req**, C_Ind_R, D_Req_R, **dis_ind**, D_Ind_E C_Req_E, **con_req**, C_Ind_R, C_Resp_R, **con_acc**, C_Conf_E, R_Req_E, **rei_req**, R_Ind_R, R_Resp_R, **rei_acc**, R_Conf_E, D_Req_E, **dis_ind**, D_Ind_R>.

Passo_3: Uma nova seqüência de pares de eventos de entrada/saída é obtida após a projeção da seqüência acima obtida (Passo 2) no alfabeto de entrada (V) da EP_E.

Exemplo: Considerando o alfabeto de entrada definido na Tabela 1, ou seja, V = {*C_Req, D_Req, con_acc, dis_ind, R_Req, rei_acc*}:

<C_Req_E/con_req, con_acc/C_Conf_E, dis_ind/D_Ind_E, C_Req_E/con_req, dis_ind/D_Ind_E, C_Req_E/con_req, con_acc/C_Conf_E, R_Req_E/rei_req, rei_acc/R_Conf_E, D_Req_E/dis_ind>

Passo_4: É satisfeita a restrição (2.1) pela inclusão, na seqüência de treinamento, de eventos escolhidos randomicamente a partir de V – {eventos esperados}. Neste caso, a EP_E responde com saída nula (*Null*). Uma exceção é feita em relação aos eventos de entrada *dis_ind* e D_Req_E. Neste caso, a EP_E deve responder com D_Ind_E e *dis_ind*, respectivamente, e o evento de entrada subseqüente deve ser a primitiva de serviço C_Req_E. Neste caso o resto da seqüência deve ser reescrito.

Exemplo: A seqüência de treinamento (*ST*) é obtida após este passo:

<C_Req_E/con_req, **C_Req_E/Null**, con_acc/C_Conf_E, **rei_acc/Null**, dis_ind/D_Ind_E, **con_acc/Null**, C_Req_E/con_req, dis_ind/D_Ind_E, **R_Req_E/Null**, C_Req_E/con_req, **dis_ind/D_Ind_E**, C_Req_E/con_req, **C_Req_E/Null**, con_acc/C_Conf_E, **con_acc/Null**, **D_Req_E/dis_ind**>.

A seqüência de treinamento acima obtida é apresentada abaixo utilizando a codificação da Tabela 1:

ST = <0/2, 0/6, 2/0, 5/6, 3/1, 2/6, 0/2, 3/1, 4/6, 0/2, 3/1, 0/2, 0/6, 2/0, 2/6, 1/3>.

B. Comprimento das Seqüências de Treinamento

As *STs* devem ser longas o suficiente para exercitar todos os caminhos da FSM que descreve a EP_E. Uma seqüência muito curta pode causar ambigüidade na descrição do comportamento desejado. Por outro lado, uma seqüência muito longa pode afetar o desempenho do sistema.

A solução para o problema de encontrar o tamanho ideal das seqüências de treinamento foi obtida da fórmula derivada em [8], baseada na solução para o problema "tempos de espera em amostragem", descrito em [14]. Esta fórmula define o comprimento L da seqüência de entrada como $L = E(S) \times E(I)$, onde $E(S)$ e $E(I)$ são os *números esperados* de transições de estado e de eventos de entrada, respectivamente, podendo ser calculados através da fórmula $E(N) = N\,(1 + {}^1/_2 + \ldots + {}^1/_N)$. Entretanto, já que o número de estados (S) necessários para descrever a especificação do protocolo é desconhecido, S deve ser sobreestimado a priori.

C. Codificação do Cromossomo

O cromossomo utilizado para codificar uma FSM utiliza representação baseada em estado. Este cromossomo, com S estados e I entradas, é mostrado na Figura 5.

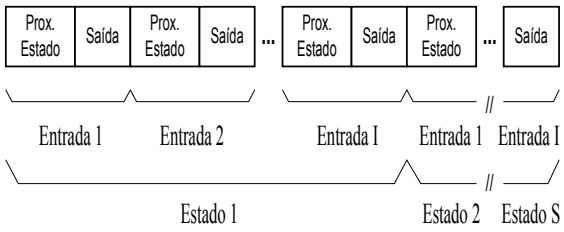

Fig. 5. Codificação do cromossomo.

Na presente metodologia, diferentemente de outras abordagens [9], o comprimento do cromossomo pode variar. A metodologia utiliza PG por ser esta a mais eficiente técnica para implementar um processo evolutivo com cromossomos de tamanho variável. A função objetivo é ponderada pelo inverso do número de estados que implementam a FSM, o que induz o processo evolutivo a buscar por máquinas de estado que atendam à funcionalidade desejada e que contenham número reduzido de estados.

D. Função Objetivo

A função objetivo é uma função com múltiplos parâmetros que o algoritmo irá maximizar. O valor de aptidão, atribuído a um dado comportamento da FSM, é ponderado pelo inverso do número de estados utilizados pela FSM. A função de aptidão F tem a forma:

$$F = \sum_{i=1}^{N} w_i H_i + \frac{W}{S} \qquad (1)$$

onde N é o número de casos de treinamento, w_i é o fator de ponderação para o caso de treinamento i, S é o número de estados da FSM, W é uma constante e H_i é o número de acertos relativo ao caso de treinamento i. H_i é calculado da seguinte forma: inicialmente a FSM é colocada no estado inicial; em seguida, para cada evento de entrada da seqüência de treinamento, as saídas da FSM são comparadas às saídas corretas das seqüências de treinamento e um acerto é assinalado em caso de igualdade.

IV. EXPERIMENTO

O objetivo deste experimento é derivar a especificação da entidade de protocolo ($EPRO_E$) do lado emissor (EP_E) de um protocolo orientado à conexão a partir de seqüências de treinamento (*STs*) geradas utilizando o método de derivação apresentado na Seção III.A.

A. Definição da BNF

Uma BNF (*Backus-Naur Form*) descreve estruturas admissíveis (sintaxe) na construção de uma linguagem através de uma 4-tupla $\{S,N,T,P\}$, onde S denota o símbolo inicial, N denota um conjunto de símbolos não terminais, T denota um conjunto de símbolos terminais e P são as produções. Uma produção, ou regra de transformação de *string*, é uma substituição do tipo $X \rightarrow Y$, onde $X \in N$ e $Y \in (N \cup T)^*$. A seguinte BNF permite a codificação de cromossomos com tamanho variável:

```
S          := <fe>;
<fe>       := "S" <expr> "E";
<expr>     := <if_stat>|<expr> <if_stat>;
<if_stat>  := <next_st><out> <next_st><out>
              <next_st><out> <next_st><out>
              <next_st><out> <next_st><out>;
<next_st>  := "0"|"1"|"2"|"3"|"4"|"5";
<out>      := "0"|"1"|"2"|"3"|"4"|"5"|"6";
```

Na BNF mostrada acima, o símbolo não terminal <*expr*> permite que o cromossomo varie seu tamanho, uma vez que o número de símbolos não terminais <*if_stat*> pode variar entre indivíduos. O símbolo não terminal <*if_stat*>, por sua vez, fixa o número de eventos de entrada (em 6, na descrição BNF acima definida, cada um produzindo um par próximo estado/saída, isto é, <*next_st*> <*out*>).

B. Atributos de PG

As principais características do sistema de PG para o problema em questão estão definidas na Tabela 2.

Tabela 2: Características de PG para a síntese de protocolos

Objetivo:	Evoluir uma FSM, com número de estados reduzidos, que descreva corretamente a $EPRO_E$, a partir de *STs*.
Símbolos terminais:	Saídas: 0, 1, 2, 3, 4, 5 e 6; Identificadores de Estado: 0, 1, 2, 3, 4, 5 e 6; Delimitadores de *string*: "S" (*Start*) e "E" (*End*).
Símbolos não-terminais:	<fe>, <expr>, <if_stat>, <next_st> e <out>.
Casos de treinamento:	Dezoito *ST* de 32 bits derivadas da *ES* e de um conjunto de PDUs associadas (Cf. Subseção 3.1).
Aptidão:	Número de acertos de eventos de saída, ponderado pelo inverso do número de estados utilizado para implementar a FSM (Cf. equação (1)).
Método de seleção:	Proporcional à aptidão.
Parâmetros principais:	M=1000, G=400, p_c=0.63, p_r=0.12, p_m= 0.25, sem elitismo.

C. Resultados

O experimento foi realizado com uma população de 1000 indivíduos que evoluiu através de 400 gerações. O comprimento das seqüências de treinamento foi calculado utilizando seis eventos de entrada e o número de estados estimado em cinco, o que resultou em seqüências com 168 pares de eventos de entrada/saída. Na prática, foram utilizadas 18 seqüências de 32 bits, o que corresponde a mais de três seqüências de 168 bits. Os demais valores dos parâmetros utilizados na função objetivo foram $W = 60$ e $w_i = 1$, já que todas as seqüências de treinamento têm o mesmo tamanho.

Em uma execução particular, um único indivíduo atendeu totalmente às restrições de entrada e saída, utilizando um número reduzido de estados. A Figura 6 mostra a máquina Mealy resultante, que descreve com sucesso a $EPRO_E$ utilizando um grafo de transição de estados (STG: *State-Transition Graph*). Nesta figura, um rótulo *v*/*v'* representa um arco e_{ij} entre dois estados q_i e q_j se, e somente se, $o(q_i,v)=q_j$ e $t(q_i,v)=v'$. Este indivíduo foi sintetizado com apenas 4 estados na geração 119, apresentando F = 591 para um total de 18 x 32 = 576 acertos. A evolução da aptidão do melhor indivíduo e da média da população está apresentada na Figura 7. Indivíduos funcionalmente corretos, embora não otimizados, foram encontrados em outras execuções. A Figura 8 mostra um desses indivíduos, com respectivo cromossomo.

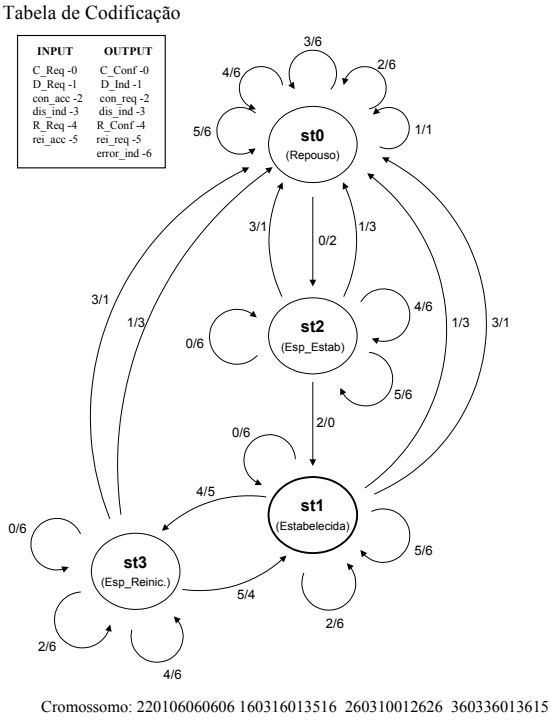

Fig. 6. $EPRO_E$, utilizando STG, do indivíduo mais apto.

D. Discussão da Correção Semântica

A priori, a correção semântica, que significa que a FSM implementa corretamente todas as funções desejadas, não pode ser garantida por uma máquina de estados que atenda a seqüências parciais de entrada e saída. No entanto, a metodologia se apóia no trabalho desenvolvido por Manovit *et al.* [8] e Chongstitvatana e Aporntewan [9] que investigaram a influência do número e do tamanho dessas seqüências no que eles denominaram *porcentagem de correção*. A porcentagem de correção foi definida como sendo a relação entre o número de execuções produzindo *soluções completas*, isto é, soluções que operam corretamente para *todas* as possíveis seqüências de entrada/saída, e o número de execuções produzindo *soluções incompletas*, ou seja, soluções que operam corretamente para as seqüências de entrada/saída *testadas*. Os autores mostraram, por meio de

experimentos, que a porcentagem de correção pode ser elevada a 100%, neste caso garantindo a correção semântica, com o aumento do comprimento das seqüências de treinamento, assim como do número de tais seqüências.

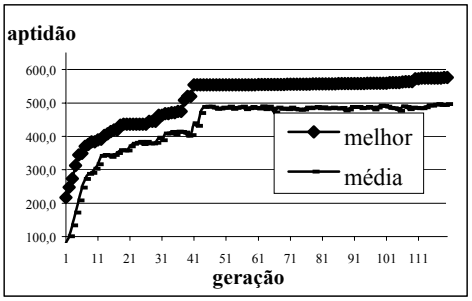

Fig. 7. Histograma de aptidão.

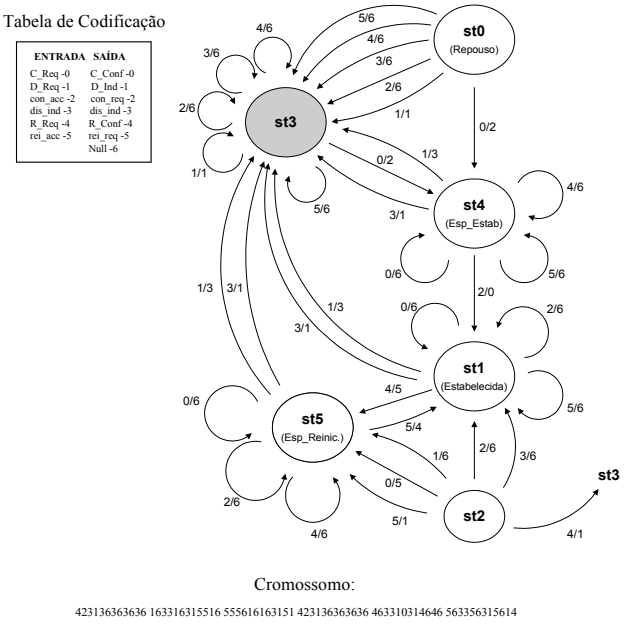

Fig. 8. Solução correta com um estado redundante (círculo sombreado) e um estado não alcançável (círculo pequeno).

O experimento acima relatado obteve sucesso no teste de equivalência fraca (*weak bissimulation*), para o qual foi utilizada a ferramenta CWB (*Concurrency Workbench*). Para este propósito, o lado receptor também foi evoluído e ambas as entidades, emissora e receptora, juntamente com o modelo global do serviço do protocolo (Figura 2a), foram traduzidas para CCS (*Calculus of Communicating Systems*).

V. CONCLUSÃO

Foi proposto um método não convencional de síntese de protocolos de comunicação, que reduz o número de etapas necessárias à geração de especificações de entidades de protocolo, elevando o nível de abstração do projetista.

O programa que implementou o sistema de programação genética mostrou-se eficiente. Uma execução, para um limite máximo de 400 gerações de uma população de 1000 indivíduos, consumiu menos de 30 segundos numa máquina com arquitetura X86 de 1GHz de relógio. Considerando que técnicas tradicionais de síntese de protocolo que utilizam FSMs [3, 6] são truncadas para a execução de várias etapas de transformações em autômatos, o tempo acima mencionado é bastante reduzido. Estas etapas são reduzidas ou eliminadas na metodologia, da seguinte forma: (i) a projeção é realizada de forma direta, (ii) a determinização é eliminada e (iii) o autômato é automaticamente reduzido como conseqüência da adoção de um processo evolutivo que contou com a utilização de cromossomos de tamanhos variáveis e com uma função de aptidão inversamente ponderada pelo número de estados.

A metodologia garante de antemão a completude da especificação do protocolo, isto é, ausência de erros devido à recepção de evento não especificado, pois gera FSMs completamente especificadas. A correção semântica é garantida aumentando-se o número e o comprimento das seqüências de treinamento. Entretanto, já que as técnicas evolutivas utilizam probabilidade, um estudo mais específico deve ser realizado com o intuito de quantificar os dois parâmetros envolvidos nesta correção (Cf. Seção IV.D). Este trabalho assumiu a restrição de comportamento estritamente seqüencial para protocolos de comunicação, que também é adotada em diversos trabalhos de pesquisa que utilizam o modelo de FSM, como em [3, 6].

REFERÊNCIAS

[1] A. Tanenbaum, "Computer Networks", 3rd edition, 1996.

[2] H. Yamaguchi, "Implementation of Service Specifications on Distributed Computing Systems", PhD thesis, Osaka Univ., 1998.

[3] A. Khoumsi, and K. Saleh, *Two Formal Methods for the Synthesis of Discrete Event Systems*, Computer Networks and ISDN Systems, Vol. 29, No. 7, pp. 759-780, 1997.

[4] K. El-Fakih, H. Yamaguchi, G. Bochmann and T. Higashino, *Protocol Re-synthesis Based on Extended Petri Nets*, in Proc. of Int. Workshop (SEPN-2000), pp. 173-188, 2000.

[5] H. Yamaguchi, K. Okano, T. Higashino and K. Taniguchi, *Synthesis of Protocol Entities' Specifications from Service Specifications in a Petri Net Model with Registers*, in Proc. of ICDCS-15, pp. 510-517, 1995.

[6] A. Khoumsi, G. Bochmann and R. Dssouli, *Protocol Synthesis for Real-Time Applications*, Joint Int. Conf. on Protocol Specification, Testing and Verification (PSTV) and FORTE, Beijing, China, 1999.

[7] K. El-Fakih, H. Yamaguchi and G. Bochmann, *A Method and a Genetic Algorithm for Deriving Protocols for Distributed Applications with Minimum Communication Cost*, in Proc. of the 11th IASTED, (PDCS'99), 1999.

[8] C. Manovit, C. Aporntewan and P. Chongstitvatana, *Synthesis of Synchronous Sequential Logic Circuits from Partial Input/Output Sequences*, in Proc. of ICES'98, pp. 98-105, 1998.

[9] P. Chongstitvatana and C. Aporntewan, *Improving Correctness of Finite-State Machine Synthesis from Multiple Partial Input/Output Sequences*, in Proc. of the 1st NASA/DoD Workshop on Evolvable Hardware, pp. 262-266, 1999.

[10] H. Hemmi, J. Mizoguchi and K. Shimohara, *Development and Evolution of Hardware Behaviors*, Int. Workshop, Lausanne, E. Sanchez and M. Tomassini (Eds.), LNCS 1062, pp. 250-265, 1995.

[11] H. Hörner, *A C++ Class Library for Genetic Programming*, Release 1.0 Operating Instructions, Viena University of Economy, 1996.

[12] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[13] R. Probert and K. Saleh, "Synthesis of Protocols: Survey and Assessment", IEEE Transactions on Computers, Vol. 40, No 4, pp. 468-476, 1991.

[14] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. I, Wiley, pp. 224-225, 1968.