PORTABILIDADE DE UM COMPILADOR ESTELLE/83 PARA A CRIAÇÃO DE UMA BASE DE PESQUISAS EM TDF'S

Edilson Ferneda (**)
Peter Eugene Lonergan (*)
Solòn Benayon da Silva (*)
Wanderley Lopes de Souza (**)

Centro Científico Rio (*)
IBM Brasil
Caixa Postal 4624
20.071, Rio de Janeiro, RJ

Grupo de Redes de Computadores (**)
Universidade Federal da Paraiba
Caixa Postal 10032
58100, Campina Grande, Pb

SUMÁRIO

Este trabalho descreve em linhas gerais o projeto que vem sendo desenvolvido em conjunto pelo Centro Cientifico Rio (CC-RIO), da IBM Brasil e pelo Grupo de Redes de Computadores (GRC), do Departamento de Sistemas de Computação da Universidade Federal da Paraiba, na área de linguagens para a especificação de serviços e protocolos de comunicação.

Em seguida é abordada a primeira fase do projeto, já concluida, que teve como objetivo adaptar o compilador ESTELLE/83, desenvolvido pelo GRC, ao ambiente computacional do CC-RIO, avaliando suas caracteristicas de portabilidade e operacionalidade.

1. INTRODUÇÃO

Em julho de 1988, o Centro Científico Rio da IBM Brasil e o Grupo de Redes de Computadores do Departamento de Sistemas de Computação da Universidade Federal da Paraíba, assinaram um Programa de Trabalho para o Desenvolvimento Conjunto de Pesquisas na Área de Comunicação de Dados, com o objetivo básico de desenvolver projetos de pesquisa na área de linguagens de especificação de serviços e protocolos de comunicação.

Este programa se insere dentro de uma linha de pesquisa mais ampla, cujo objetivo principal é a criação de grupos de competência, não somente na área de linguagens de especificação e suas aplicações mas também na área de validação e simulação de protocolos de comunicação. Assim é que a idéia inicial objetiva criar uma base de pesquisas, no Brasil, que coloque disponível, as ferramentas necessárias para o desenvolvimento de trabalhos nas principais linguagens de especificação de protocolos.

Estão previstos os estudos e a implantação de ferramentas baseadas nas seguintes linguagens: ESTELLE (ISO 86a), LOTOS (ISO 86b), SDL (SaTi 87) e PASS (IBM 88).

Após a criação dessa base de pesquisas, é intenção se ter a criação de outros programas conjuntos de pesquisas, com entidades acadêmicas e científicas, de tal forma a, tendo como ponto de partida essa base inicial, partir para uma especialização nas diversas linguagens, de acordo com a preferência e tendência de cada entidade interessada.

O Programa de Pesquisas teve como ponto de partida o projeto que já vinha sendo desenvolvido pelo GRC, na área de linguagens de especificação de protocolos, mais precisamente, nos trabalhos do GRC específicos sobre a linguagem ESTELLE relativos a criação de um compilador para aquela linguagem.

Visando já a criação da base comum de pesquisas e, para uma avaliação de sua portabilidade e consequente oportunidade de difusão, foi escolhido como uma primeira fase do programa conjunto de pesquisas entre o GRC e o CC-RIO, a adaptação do Compilador ESTELLE/83, desenvolvido na UFPb, para o ambiente computacional do Centro Científico Rio.

Como atividade paralela ao programa, vem o Centro Científico Rio implantando também, em seu ambiente computacional, ferramentas de implementação do PASS (Parallel Activity Specification Scheme), uma técnica para a especificação e a implementação de processos paralelos, técnica essa baseada em uma máquina de estados finita estendida.

A primeira fase teve início em julho de 1988, tendo uma duração total de dois meses. Esta fase foi composta das seguintes atividades:

- avaliação e estudos iniciais sobre o Compilador ESTELLE/83

  Nesta atividade foram mapeadas as necessidades computacionais, de hardware e software, para a instalação do Compilador em uma configuração diferente da inicialmente utilizada.
- avaliação e estudos sobre as linguagens de programação utilizadas no Compilador.

  O Compilador ESTELLE/83 foi desenvolvido pelo GRC tendo como linguagem básica de programação uma versão de PROLOG conhecida como

Waterloo/PROLOG. No ambiente computacional do CC-RIO, a linguagem utilizada corresponde a uma versão conhecida como VM/PROLOG, daí a necessidade de se fazer uma conversão entre as versões da linguagem PROLOG.
- Conversão de Waterloo/PROLOG para VM/PROLOG, avaliação e análise das dificuldades encontradas.
- Instalação e testes na nova versão do Compilador ESTELLE/83 no novo ambiente
- Documentação e Divulgação dos resultados

No próximo bloco é descrito o compilador ESTELLE/83, nos seus aspectos arquiteturais, aspectos de construção bem como os testes realizados tanto no ambiente inicial como no novo ambiente, de tal forma a se ter uma base de comparação para avaliar as dificuldades no seu transporte para um ambiente diferente daquele para o qual foi inicialmente projetado.

2. O COMPILADOR ESTELLE/83

Na época em que foram iniciados os trabalhos relativos ao compilador ESTELLE/83 (1985), as TDFs Estelle e LOTOS estavam em franco desenvolvimento e evolução. Na ocasião, optou-se por utilizar uma versão precursora (ISO 83) da TDF ESTELLE para a construção desse compilador(FERN 88). Para a construção de uma implementação completa, a partir de uma especifição em ESTELLE/83 (BoGe 84) foram definidas as seguintes etapas (Figura 1):

1. compilação da especificação formal
2. desenvolvimento de um núcleo de exploração
3. codificação das rotinas para a implementação das primitivas de baixo nível, responsáveis pela utilização dos recursos oferecidos pelo ambiente computacional.

As implementações geradas, a partir da especificação em ESTELLE/83, se constituem de estruturas de dados e de procedimentos que devem, em princípio, operar simultaneamente. Na maioria das vezes, porém, esse paralelismo não é real mas simulado. Por esta razão, foi implementado um núcleo, responsável pela execução pseudo-paralela dos processos. Esse núcleo tem como objetivos (FeLo 86):

1. definir uma política de escalonamento para a recepção/emissão de mensagens;
2. identificar e acionar os processos envolvidos no intercâmbio de mensagens e
3. gerir as transições espontâneas (internas).

Uma vez que a implementação das primitivas de baixo nível depende quase que exclusivamente do ambiente em que atuam, elas foram codificadas a parte numa linguagem apropriada.

Além disso, foram criadas rotinas de suporte, para a manipulação das estruturas de dados geradas pelo compilador.

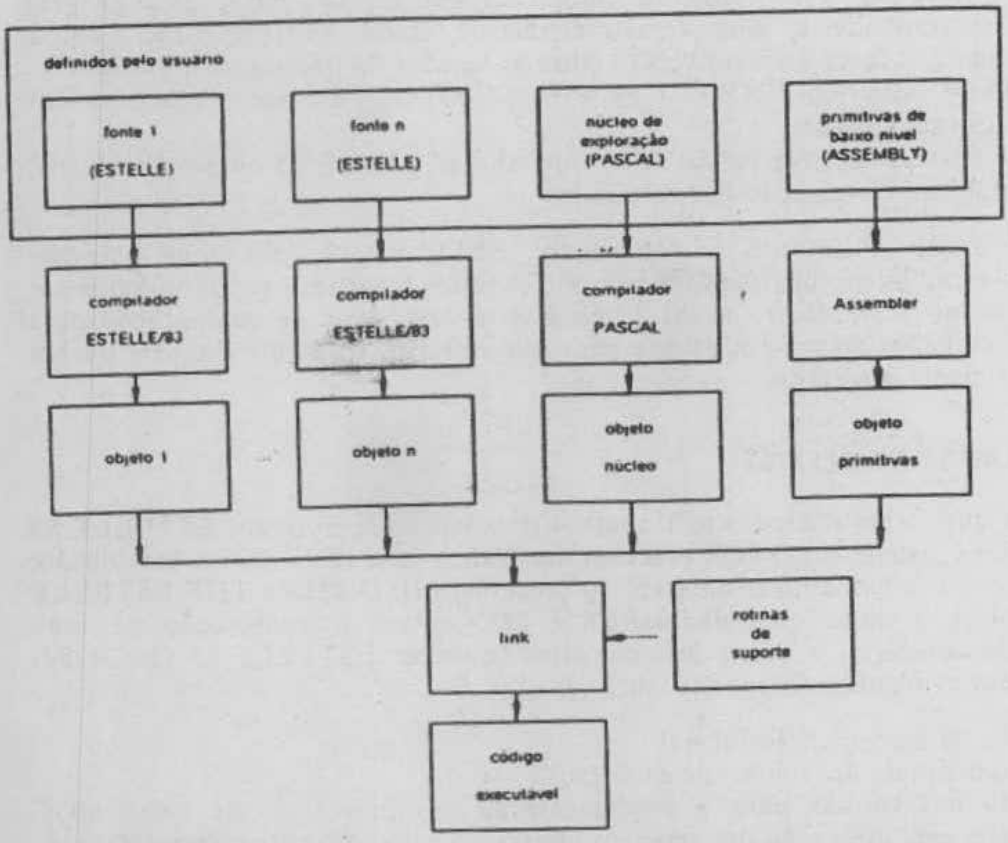

Figura 1 - Etapas e procedimentos necessários para alcançar um código executável

3. A ARQUITETURA DO COMPILADOR

Um sistema especificado em ESTELLE/83 é geralmente composto de vários módulos "fontes", que interagem através de mensagens. É necessário portanto, que o compilador trate esses módulos separadamente.

O compilador ESTELLE/83 utiliza a sintaxe apresentada em (ISO 83) e gera, como um passo intermediário, um programa numa linguagem de alto nivel. Esse programa é constituido de um conjunto de estruturas de dados e de um conjunto de procedimentos, que refletem os conceitos próprios da TDF ESTELLE/83. Uma vez que essa TDF tem como suporte a linguagem de programação PASCAL, torna-se natural o uso de PASCAL (ou outra linguagem correlata) como linguagem de implementação.

O compilador ESTELLE/83 é constituido de um pré-processador e de um compilador PASCAL. A função desse pré-processador é traduzir as construções próprias à TDF ESTELLE/83, produzindo um programa totalmente em PASCAL (Figura 2).

O pré-processador ESTELLE/83 é responsável pela análise do código fonte ESTELLE/83 e pela geração do correspondente código PASCAL da implementação.

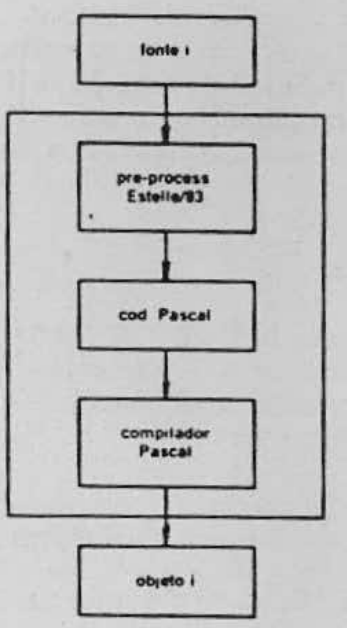

Figura 2 - Estrutura do compilador ESTELLE/83

Na atividade de análise podem ser distinguidas quatro funções principais (Grie 71):

1. análise léxica, responsável pela leitura da especificação, em ESTELLE/83, e pelo reconhecimento de cada item léxico dentro dessa especificação;
2. análise sintática, que tem como fonte os itens léxicos gerados pelo analisador léxico, agrupa-os em diversas unidades sintáticas. A verificação da pertinência (ou não) dessas unidades sintáticas é realizada com base na BNF de ESTELLE/83;
3. ações semânticas, acionadas pelo analisador sintático para a criação de uma representação interna para as estruturas sintáticas ESTELLE 83 e
4. análise de contexto, acionado pelas ações semânticas, para a verificação da correção semântica dessas estruturas internas.

As funções (1), (2) e (4) verificam, respectivamente, os erros léxicos, sintáticos e de contexto. Na ocorrência de qualquer tipo de erro, uma mensagem apropriada é emitida ao usuário e a execução do compilador é interrompida. A metodologia escolhida, para o tratamento de erros, é a mais simples possível: a cada erro sinalizado, cabe ao usuário efetuar as correções apropriadas na especifição fonte, reiniciando em seguida todo o procedimento de compilação.

Uma vez terminada a atividade de análise, tem início a atividade de geração de código. Nessa etapa, duas funções principais podem ser distinguidas:

- a geração de código transforma a representação interna, gerada durante a atividade de análise, numa estrutura próxima à linguagem de implementação (PASCAL) e

- a gravação de código gera um segmento de programa PASCAL livre de erros léxicos e sintáticos.

O pré-processador realiza somente a análise de contexto, relativa às construções introduzidas pela TDF ESTELLE/83. A análise de contexto, relativa às construções da linguagem PASCAL, é responsabilidade do compilador Pascal. A deteção desse tipo de erro, no código de implementação, não cria maiores dificuldades, para a sua correção, a nivel da especificação fonte, já que essa última está refletida na implementação gerada.

4. CONSTRUÇÃO DO COMPILADOR

A linguagem PROLOG ("Programming in Logic") (ClMe 81) vem sendo utilizada na construção de ferramentas que auxiliam a especificação formal, a verificação, a implementação e os testes de protocolos. Exemplos dessas ferramentas podem ser encontrados em (BoDs 86), (JaMo 86) e (BaSa 87).

O pré-processador ESTELLE/83 foi totalmente desenvolvido em PROLOG. A escolha dessa linguagem se deu devido às seguntes vantagens (Warr 80):

- a atividade de programação em PROLOG é simples, não sendo necessário muito tempo para a obtenção de um produto final;
- a manutenção e/ou extensão de um programa escrita em PROLOG pode ser também muito simples;
- há um mapeamento direto entre a função de análise sintática e sua implementação em PROLOG e
- A linguagem PROLOG tem sido usada satisfatoriamente na construção de ferramentas de simulação para protocolos de comunicação (futuramente o compilador ESTELLE/83 deverá ser integrado a um sistema desse tipo).

A fim de verificar a viabilidade do uso de PROLOG, na construção do pré-processador ESTELLE/83, o analisador sintático foi desenvolvido, inicialmente, em micro-PROLOG (Clar 84). Apesar da constatação (LoFe 86) das potencialidades de PROLOG, o micro-PROLOG não supriu certas necessidades. O maior problema encontrado foi a limitação em relação à memória reservada para área de trabalho (o micro-PROLOG impõe 64K). Optou-se, então, pelo interpretador Waterloo PROLOG, usado num computador IBM 4341 em ambiente CMS com a reserva de 3Mb de memória para cada máquina virtual.

função	linhas de código	numero de clausulas	número de procedimentos
gerenciamento	80	9	7
análise sintática	1200	288	124
análise léxica	400	86	34
ações semânticas	650	102	76
análise de contexto	500	137	45
geração de código PASCAL	1200	227	109
gravação de código PASCAL	1000	228	98
	5030	1077	493

Figura 3 - Distribuição do código do Compilador ESTELLE/83

O programa em Waterloo/PROLOG do pré-processador ESTELLE/83 contém, aproximadamente, cinco mil linhas de código e a Figura 3 nos mostra a sua distribuição.

Deve-se salientar que 1050 linhas da análise sintática e 650 linhas da gravação de código são, respectivamente, a transcrição direta para Waterloo/PROLOG das regras da BNF de ESTELLE/83 e da BNF do subconjunto PASCAL utilizado.

Existe pouca bibliografia referente ao desenvolvimento, em PROLOG, de compiladores da magnitude do ESTELLE/83 (Moni 84). A facilidade encontrada na transcrição, em Waterloo/PROLOG, das funções do pré-processador ESTELLE/83 pode ser contraposta ao tempo, relativamente longo, dispensado na busca de soluções para os seguintes problemas:

- encontrar um tratamento adequado, para o analisador sintático, dos nós terminais das regras de produção;
- encontrar uma estrutura adequada para as representações internas ESTELLE/83 e PASCAL e, principalmente,
- encontrar um conjunto de regras adequado para a transformação da representação interna ESTELLE/83 na representação interna PASCAL (função geração de código).

5. TESTES REALIZADOS COM O COMPILADOR ESTELLE/83

Dois tipos de análise podem ser realizados com as especificações de protocolos (Boch 87):

- estática, que permite a validação léxica e sintática da especificação. Permite também a validação de alguns aspectos semânticos, tais como os erros de contexto, relativos aos conceitos arquiteturais da TDF utilizada (análise semântica estática).
- dinâmica, que permite investigar o comportamento da especificação, submetendo-a a um conjunto de cenários possíveis. A análise dinâmica pode ser realizada de forma exaustiva ou através de simulação. No primeiro caso, são considerados todos os cenários possíveis. Na simulação, apenas alguns desses cenários são considerados.

Cabe salientar que os tipos de erros, detectáveis através da análise dinâmica, não podem ser percebidos na análise estática, pois esta última não leva em consideração o ambiente distribuído no qual o protocolo foi inserido.

Nos testes do Compilador ESTELLE/83 foram efetuadas uma análise estática e duas análises dinâmicas (por simulação) de uma especificação relativa a um protocolo do tipo bit-alternante. A Figura 4 descreve os módulos, utilizados na especificação desse protocolo, as mensagens, utilizadas para as interações entre os módulos, e as primitivas de baixo nível, utilizadas para as interações entre esses módulos e o ambiente no qual esse protocolo foi inserido.

A análise estática da especificação do protocolo foi feita através de sua compilação sucessiva, até que todos os erros léxicos, sintáticos e de contexto foram erradicados. Um código executável foi alcançado através da ligação ("linking") dos seguintes objetos:

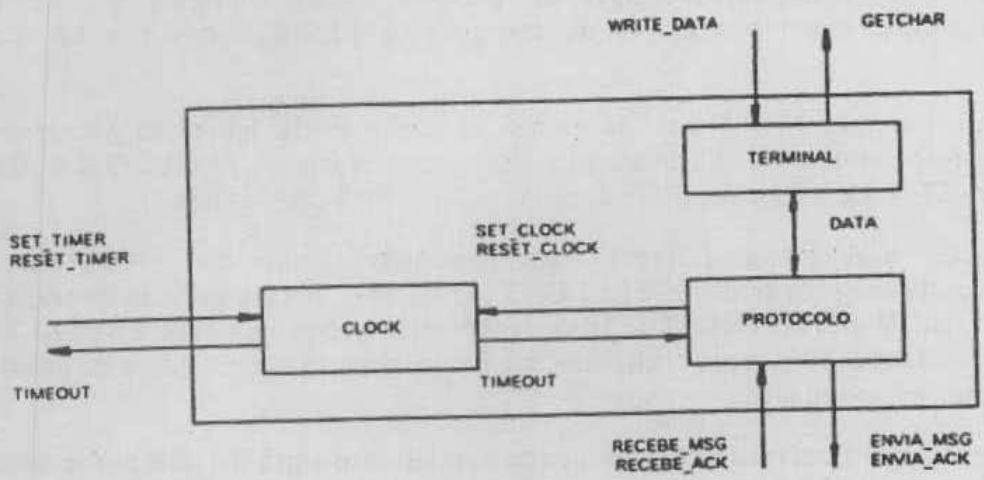

Figura 4 - Descrição dos modulos e das interações
(mensagens e primitivas de baixo nivel) do protocolo exemplo.

1. o código gerado pelo compilador PASCAL, a partir da implementação do protocolo;
2. o código de um núcleo de exploração, adaptado para essa implementação;
3. o código das rotinas de suporte, que auxiliam na construção e manipulação das estuturas de dados, que por sua vez foram definidas pela implementação e
4. o código das primitivas de baixo nivel.

No compilador ESTELLE/83, parte da análise semântica estática foi feita pelo compilador PASCAL. Para a análise dinâmica, optou-se pela especificação de um sistema composto de duas ocorrências do protocolo, interagindo através de um módulo, que por sua vez simula o meio de comunicação (Figura 5).

Nessa especificação, as primitivas de baixo nivel, trocadas entre os submódulos PROTOCOLO e o ambiente computacional, foram substituidas por mensagens, agora trocadas entre os submódulos PROTOCOLO e o módulo MEIO_DE_TRANSMISSÃO.

Dois tipos de meio foram especificados: A primeira análise considerou um meio livre de perdas e de atrasos de transmissão e a segunda análise dinâmica considerou um meio de transmissão sujeito a perdas de informação. Em ambas as simulações o protocolo comportou-se da forma esperada, outorgando assim um certo grau de confiabilidade à sua especificação.

Para que se possa realizar uma avaliação mais contundente do compilador ESTELLE/83 é necessário especificar e analisar um protocolo mais complexo (e.g.: os relativos ao modelo OSI). O tempo para se chegar a uma especificação completa em ESTELLE/83 e para se obter uma implementação semi-automática confiável de um protocolo de grande porte, é relativamente longo. Essa avaliação está sendo realizada durante os trabalhos de desenvolvimento dos protocolos Transporte (classe 2), Sessão

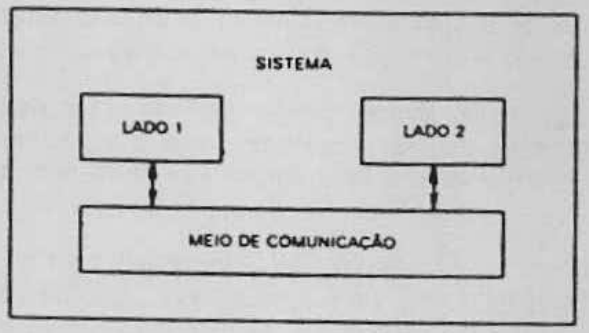

Figura 5 - Módulos do sistema a ser simulado para a análise dinâmica do protocolo exemplo.

(subconjunto BCS) e FTAM (núcleo), objetivos de fases posteriores do programa conjunto de pesquisas.

6. CONVERSÃO, INSTALAÇÃO E TESTES PARA A VERSÃO VM/PROLOG

Existem várias implementações de PROLOG, dentro de uma linha considerada oficiosamente como "padrão", podendo ser citadas dentre elas o VM/PROLOG, o Waterloo/PROLOG e o ARITY. Essas implementações são essencialmente equivalentes, diferenciando-se somente por algumas opções de projetos dos seus extra-lógicos (writes, reads, predicados de controle etc.).

Esta similaridade já não se verifica com outras versœs de PROLOG que fogem dessa linha "padrão", tais como as versœs denominadas NU-PROLOG, MI-PROLOG etc.. Por esta razão, essas versœs não foram consideradas quando do estudo de portabilidade para o compilador ESTELLE/83.

O Compilador ESTELLE/83 foi inicialmente escrito em Waterloo/PROLOG por ser essa a lingagem disponivel a época de seu desenvolvimento, na UFPb. Ele foi instalado e testado no ambiente computacional do Campus I da Universidade Federal da Paraiba. Este ambiente era composto por uma Unidade Central de Processamento modelo IBM 4341 com um Sistema Operacional VM/CMS. Para a instalação do Compilador e de seus programas de teste foi criada uma máquina virtual com memória de 3 Megabytes.

Para testes do compilador foi especificada uma versão do protocolo de bit alternante que resultou em 250 linhas de código em ESTELLE/83.

O tempo decorrido para a obtenção do código executável girou em torno de três minutos, com o uso dedicado da U.C.P.. Esse tempo, relativamente grande, se deve principalmente aos tempos gastos nas operações entrada/saida e também as características de interpretação do Waterloo/PROLOG para a obtenção do codigo PASCAL.

Outro problema encontrado foi a falta de otimização no uso da memória na Unidade Cantral de Processamento. A versão Waterloo/PROLOG não dispõe de funções que permitam a "coleta de lixo", que vem a ser a capacidade de liberar áreas de memória que já foram utilizadas e não são mais necessárias. Essa deficiência faz com que a execução se torne cada vez mais lenta, a medida em que áreas de memória disponiveis vão diminuindo. Além disso essa limitação também proibe a implementação de protocolos mais complexos devido a limitação prática no uso da memória virtual.

O Centro Científico Rio da IBM Brasil dispõe de uma Unidade Central de Processamento modelo IBM 4381 de 32 Megabytes com um Sistema Operacional VM/HPO versão 4.2. Cada usuário dispõe de máquinas virtuais com memória de até 6 Megabytes.

Como primeiro passo para a conversão do Compilador ESTELLE/83, de Waterloo/PROLOG para VM/PROLOG, foi feita uma avaliação das diferenças entre os comandos usados. Dessa avaliação resultaram as diferenças mostradas na Figura 6, o que mostrou não ser necessário grande esforço de programação na realização da conversão, que foi feita por um pesquisador e um programador de sistemas. Esses dois profissionais dispenderam cerca de 40 horas de trabalho, incluindo-se ai não somente a conversão mas também a parte de teste e documentação.

Waterloo-Prolog	VM/Prolog
'asd' e' string ou constante	'asd' e string e "asd" e constante
writech( )	prst( )
Obs.: writech('.') & writech(X) & writech('.')	prst('.'!!X!!'.')
string(A,B)	st_to_li(A,B)
string(X,'a'.'0'.*)	string(X,"a".Z.*) & eq(Z,0)
string('0',X)	st_to_at(A,'0') & string(A,X)
newline	nl
a <-	a <- b
addax(a,N)	addax(a) & ultima(a,N)
delax(a(X),N)	deleta(a(X),N)
stop	fin
prolog 2640	vmprolog rule 2640k
load(estelle).	<- include(est,new).
	<- est.
read(X,in) e write(X,out)	dcio(arqin,ioput,file,in,prolog,a,f,80)
	dcio(arqout,output,file,out,prolog,a,f,80)
	dcio(arqin,close)
	dcio(arqout,close)

Figura 6 - Diferenças entre o Waterloo/PROLOG e o VM/PROLOG

Devido ainda aos tipos de diferenças apresentadas, o número de linhas de código não sofreu grande alteração, tendo sido observado um aumento que não ultrapassou 10% do total de linhas. Isso fez com que a versão VM/PROLOG necessitasse uma área de memória maior do que a versão Waterloo/PROLOG, sendo necessário a definição de uma máquina virtual de 3,5 Megabytes.

Com a nova versão, e utilizando a mesma especificação do protocolo de bit alternante, foram executados os mesmos testes realizados com a versão original tendo sido obtidos resultados de natureza idêntica.

Com relação ao desempenho do Compilador, não foram observadas diferenças significativas uma vez que também o VM/PROLOG é uma linguagem interpretada. Melhoras significativas no desempenho poderiam ser observadas caso fosse utilizada alguma versão compilada.

Ponto significativo para a versão VM/PROLOG, foi a observação do funcionamento da função "coleta de lixo" que permitiu uma otimização significativa na utilização de áreas de memória de trabalho. Esse fato vai permitir agora utilizar a ferramenta na especificação de protocolos mais complexos, o que é o objetivo da segunda fase do projeto de pesquisas.

7. CONCLUSÕES E TRABALHOS SEGUINTES

A maior razão para a escolha do PROLOG na construção do pré-processador ESTELLE/83, foi a de facilitar sua adaptação para um futuro padrão da TDF ESTELLE. Para avaliar essa adaptação, tomou-se como base uma versão mais recente de ESTELLE, apresentada em (ISO 86a) e, a partir da análise realizada em (Linn 86) dessa versão, foram constatadas grandes diferenças com relação a versão ESTELLE/83. Como exemplo dessas diferenças pode ser citada a possibilidade de criação dinâmica de módulos.

Essas diferenças nos levam a concluir que a a adaptação do pré-processador ESTELLE/83 à uma futura versão padrão do ESTELLE não será trivial. No entanto, essa adaptação seria praticamente impossível se tivessemos optado por uma linguagem procedural na construção do compilador ESTELLE/83.

Com relação ao desempenho, tanto da versão Waterloo/PROLOG como da versão VM/PROLOG, acredita-se que a implementação, em uma linguagem procedural, de certas funções do pré-processador ESTELLE/83 (por exemplo, a análise léxica), reduzirá consideravelmente o tempo gasto na obtenção do código executável, desde que essas funções possam ser colocadas em módulos independentes.

Como recomendação imediata foram sugeridas algumas modificações relativas a linguagem, modificações essas fáceis de serem implementadas e que, embora não significando grandes melhoras, vão permitir uma melhor estruturação do sistema. Essas modificações são as seguintes:

- Correção da geração de código para as transições com a clausula 'any';
- Analisador léxico em Pascal
- Substituição de alguns predicados pré-definidos, do Waterloo-Prolog, por outros, do VM/Prolog mais apropriados como por exemplo, substituição do "string(A,B) & string(C,"A"."0"B)" por "stconc('A0',A,C)"
- aglutinação de alguns predicados através do uso de '!'. Por exemplo, substituição de:
  - s18(X) <- append(*,Y.nil,X) & type_procfunc(nil,Y.' = '.A.'\\.*) & tipo(A,"S") & string(A,"'".*.C.*) & ne(C,"'") & erro_semantico(10) & /.
  - s18(X) <- append(*,Y.nil,X) & type_procfunc(nil,Y.' = '.A.'\\.*) & tipo(A,"S") & string(A,"'"."'".*) & erro_semantico(10) & /.

  por
  - s18(X) <- append(*,Y.nil,X) & type_procfunc(nil,Y.' = '.A.'\\.*) & tipo(A,"S") & ((string(A,"'".*.C.*) & ne(C,"'")) ! string(A,"'"."'".*)) & erro_semantico(10) & /.

- qualquer outro recurso do VM/Prolog que venha a melhorar a performance do compilador Estelle/83, uma vez que o uso do 'coletor de lixo' por um lado permite o processamento de especificações de qualquer tamanho, por outro lado é mais demorado
- geração efetiva dos efeitos semânticos da cláusula 'priority'
- modificação do código gerado pelas cláusulas de transição, incluindo a dependência hierárquica de cláusulas

Outra recomendação sugerida para melhorar a produtividade do pré-processador ESTELLE/83, é a definição de uma estratégia mais eficiente para a recuperação de erros (FiMi 80) e a conseqüente inclusão, ao pré-processador, dos procedimentos em PROLOG para implementar essa estratégia.

Estudos preliminares indicam que o esforço dispendido no desenvolvimento de uma implementação completa, a partir da compilação de uma especificação formal (no caso de ESTELLE/83) é cerca de 50% menor que o esforço dispendido no desenvolvimento de uma implementação manual, a partir da mesma especificação. Foi deixado para o futuro a avaliação do código PASCAL gerado, quando comparado a uma implementação desenvolvida manualmente.

Além de fornecer implementações semi-automáticas, o compilador ESTELLE/83, devido às caracteristicas intrínsicas da TDF ESTELLE/83, também fornece meios para a especificação de ambientes de simulação. Portanto, esse compilador pode ser a base para o desenvolvimento de simuladores amigáveis, que visem a validação e/ou análise de desempenho de especificações ESTELLE/83 de protocolos.

Como trabalhos futuros, o programa prevê que, já na fase seguinte a esta primeira, já desenvolvida, seja especificado um protocolo de transporte (classe 2), um protocolo de sessão (subconjunto BCS) e um protocolo FTAM (núcleo), tendo como base as ferramentas já desenvolvidas.

É ainda idéia do grupo, tendo como paradigma a descrição desses protocolos, fazer suas especificações em outras TDF's de tal forma a se chegar a um estudo comparativo entre as diversas TDF's disponiveis.

8. REFERÊNCIAS

(BaSa 87) M. Barbeau, B. Sarikaya, "CAD-PT: A computer-aided design tool for protocol testing", Relatório técnico, Concordia Univeristy, Faculty of Engineering and Computer Science, Department of Electrical Engineering, Montréal (Canadá), 1987.

(Boch 87) G.v. Bochmann, "Usage of Protocol Development Tools: The Results of a Survey", IFIP Protocol Specification, Testing and Verification: VII - An International Symposium, Participant's Proceedings, Zurique (Suiça), maio 1987

(BoDs) G.v. Bochmann, R. Dssouli, W. Lopes de Souza, B. Sarikaya, H. Ural, "Use of PROLOG for Building Protocol Design Tools", Protocol Specification, Testing, and Verification, V, editado por M. Diaz, North-Holland, Amsterdam (Holanda), 1986, pp. 131-145.

(BoGe 84) G.v. Bochmann, G. Gerber, J.M. Serre, "Semi-automatic Implementation of Communication Protocols", Département d'Informatique et de Recherche Opérationelle, Université de Montréal, Montréal (Canadá), publicação n. 518, Dezembro 1984.

(Clar 84) K.L. Clark, "micro-PROLOG: Programming in Logic", Prentice-Hall International, 1984.

(ClMe 81) W.F. Clocksin, C.S. Mellish, "Programming in PROLOG", Springer-Verlag, Berlin (Alemanha), 1981.

(FeLo 86) E. Ferneda, W. Lopes de Souza, "Compilador para a Linguagem de Especificação de Protocolos ESTELLE", Anais do XIII Seminário Integrado de Software e Hardware do VI Congresso da Sociedade Brasileira de Computação, Recife (PE), julho 1986, Vol. 1, pp. 420-428

(Fern 88) E. Ferneda, "Um compilador para a técnica de descrição formal ESTELLE/83", tese de mestrado relativa ao Curso de Mestrado em Sistemas e Computação, DSC/CCT/UFPb, Campina Grande (Pb), defendida em 09/05/88, 183 pags.

(FiMi 80) C.N. Fisher, D.R. Milton, S.B. Quiring, "Efficient LL(1) Error Correction and Recovery Using Only Insertions", Acta Informatica, vol. 13, 1980, pp. 141-154

(Grie 71) D. Gries, "Compiler Construction for Digital Computers", John Wiley & Sons, Inc., New York (EUA), 1971.

(IBM88) IBM Tech. Rep 43.8715, "PASS - The Parallel Activity Specification Scheme", European Networking Center, 1988.

(ISO 83) ISO TC97/SC16/WG1 subgroup B, "A FDT Based on an Extended State Transition Model", Working Document, Maio 1983.

(ISO 86a) ISO DP 9074, "ESTELLE - A Formal Description Technique Based on Extended State Transition Model", Outubro 1986.

(ISO 86b) ISO DP 8807, "LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", Julho 1986.

(JaMo 86) C. Jard, J.F. Monin, R. Groz, "Experience in Implementing X.250 (a CCITT subset of ESTELLE) in Veda", Protocol Specification, Testing, and Verification V, editado por M. Diaz, North-Holland, Amsterdam (Holanda), 1986, pp. 315-331.

(Moni 84) J.F. Monin, "Ecriture d'un Compilateur 'reel' en PROLOG", Centre National d'Etudes de Télécommunications, Note Technique NT/LAA/SLC/179, Lannion (França), julho 1984. (Também publicado em Journées sur la Programmation en Logique, Plestin (França), abril 1984)

(Linn 86) R.J. Linn Jr., "The Features and Facilities of ESTELLE", Protocol Specification, Testing, and Verification V, editado por M. Diaz, North-Holland, Amsterdam (Holanda), 1986, pp. 271-296.

(LoFe 86) W. Lopes de Souza, E. Ferneda, "Analisador Sintático em PROLOG para a Linguagem de Especificação ESTELLE", Anais do IV Simpósio Brasileiro de Redes de Computadores, Recife (PE), março 1986, pp. 325-353.

(SaTi 87) R. Saraco, P.A.J. Tilanus, "CCITT SDL: Overview of the Language and its Applications", Computer Networks and ISDN Systems, Vol 13, no 2, 1987.

(Warr 80) D.M.D. Warren, "Logic Programming and Compiler Writing", Software - Practice and Experience, vol. 10, 1980, pp. 97-125