BDDmeter— Uma Ferramenta para Visualização Dinâmica de BDDs

Sérgio Queiroz de Medeiros
David Déharbe
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
{sergio,david}@consiste.dimap.ufrn.br

Resumo

BDDs formam uma família de estruturas de dados utilizadas em diversos tipos de ferramentas de apoio à engenharia de sistemas computacionais como a verificação formal de *software*. BDDmeter é uma extensão gráfica para bibliotecas de BDDs a partir da qual é possível monitorar o comportamento de BDDs com o decorrer do tempo. BDDmeter possibilita em particular visualizar, sob forma de gráfico ou de texto, informações relevantes para o usuário das ferramentas baseadas em BDDs, a fim de entender melhor eventuais gargalos ocorrendo ao decorrer da execução.

1 Introdução

Os Diagramas de Decisão Binária, ou BDDs [Bry86], são uma estrutura de dados amplamente utilizada nos dias de hoje para modelagem e verificação de sistemas digitais. Os BDDs são grafos acíclicos direcionados, compostos por nós intermediários, nós terminais e arcos. Além disso, ao percorrer um determinado caminho em um BDD, a ordem total em que as variáveis são encontradas é única.

Os BDDs representam expressões booleanas de uma forma bem mais eficiente do que outras estruturas, como as tabelas verdades [BRB90], reduzindo custos na sua manipulação. Isso explica o amplo uso de BDDs em áreas como a verificação simbólica de modelos.

O tamanho do BDD depende da ordem em que estão dispostas as variáveis. No pior caso, o diagrama apresenta um tamanho exponencial com relação ao número de variáveis de entrada. Nestas circunstâncias, as operações com os BDDs acabam tornando-se impraticáveis. Vários autores [Rud93, ATB94, KF98] vêm apresentando formas de resolver/solucionar o problema do tamanho dos BDDs, visto esse ser um problema crucial, pois ele determina não somente o requerimento de memória, mas também a quantidade de tempo de execução para sua manipulação.

Para usuários de bibliotecas de BDDs um outro problema encontrado é verificar de forma precisa a evolução do tamanho de um dado BDD, como também da ordem das variáveis. Em outras palavras, ele não sabe exatamente como está crescendo o tamanho do BDD, nem pode acompanhar facilmente o comportamente de uma variável específica ao longo do tempo.

Nossa ferramenta, BDDmeter, é um plug-in visual para bibliotecas de BDDs. BDDmeter mostra graficamente em tempo real vários dados relacionados a um BDD, tais como o número de nós de cada variável, o tamanho total do BDD, além de outras informações que objetivam auxiliar usuários destas bibliotecas.

2 Conceitos Fundamentais

Um BDD é um grafo acíclico, direcionado, composto por nós intermediários, nós terminais e arcos, utilizado para representar expressões booleanas. A representação de funções lógicas por BDDs é baseada na expansão de Shannon e os algoritmos associados usam um esquema recursivo baseado nas propriedades distributivas da expansão de Shannon.

Os nós terminais de um BDD são rotulados com os valores 0 e 1. Ao passo que os nós intermediários são rotulados por uma proposição atômica p_i da função booleana f, e cada nó intermediário possui dois subBDDs associados a ele, o subBDD esquerdo e o subBDD direito, que podem ser alcançados pelo *arco 0* e pelo *arco 1*, respectivamente.

A função booleana representada por um BDD é definida pelas seguintes regras:

$$\begin{aligned} F(0) &= \text{falso} \\ F(1) &= \text{verdadeiro} \\ F(d) &= (p_i \wedge F(dir)) \vee (\neg p_i \wedge F(esq)) \end{aligned}$$

As duas primeiras equações definem a função representada pelos nós terminais, enquanto a terceira define a função representada por um nó não-terminal d rotulado com p_i e com subBDDs esq e dir.

Cada operador lógico (negação, conjunção, disjunção, etc.) é implementado através de uma operação sobre BDDs, sendo que as operações entre BDDs envolvendo operadores lógicos simples, como conjunção, são realizadas em um tempo linear com relação ao tamanho dos BDDs.

Podemos conseguir que um BDD seja uma representação canônica de uma dada expressão se fizermos a restrição que as variáveis aparecem no BDD sobre uma ordem fixa. Esta simples restrição no ordenamento das variáveis provê importantes propriedades para os BDDs, que alguns autores preferem chamar, a partir deste momento, de *OBDDs*.

Em busca de encontrar uma ordenação das variáveis que minimize o tamanho do BDD para uma dada função costuma-se usar uma técnica conhecida como ordenação dinâmica das variáveis [Rud93], na qual a ordem das variáveis é trocada em um determinado ponto da execução do programa.

p1	*p2*	*p3*	*f*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

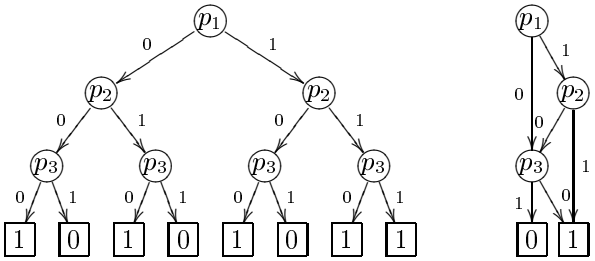

Figura 1: Representações da função $(p_1 \wedge p_2) \vee \neg p_3$ utilizando tabela verdade, árvore de decisão binária e um BDD.

A figura 1 ilustra a representação de uma mesma função booleana através de tabela verdade, árvore de decisão e BDD.

3 Descrição da Ferramenta

3.1 Padrão de Arquitetura usando BDDmeter

O padrão de arquitetura de BDDmeter pode ser entendido a partir da figura 2. Note-se que a aplicação cliente faz uso das funções da biblioteca de BDDs de forma usual, e quando é solicitada alguma função relacionada com a interface gráfica, esta chamada é repassada para a biblioteca gráfica. Foram acrescentadas à biblioteca três funções ligadas à interface gráfica:

`bdd_gui_init` dispara a criação da interface gráfica, que é executada por um novo *thread* que acessa, em modo leitura, os dados da biblioteca de BDDs;

`bdd_gui_message` permite à aplicação cliente da biblioteca de BDDs mandar imprimir mensagens para o usuário através da interface gráfica;

`bdd_gui_quit` interrompe a execução do *thread* executando a interface gráfica.

Os *threads* [Sun96] foram implementados com a biblioteca POSIX (pthreads), e a interface gráfica com a biblioteca Qt [Tro00]. Ambas bibliotecas são disponíveis gratuitamente em um grande número de plataformas.

Modificações para incluir BDDmeter em aplicações clientes de uma biblioteca de BDDs são mínimas, sendo suficiente incluir apenas chamadas às funções de inicialização e finalização da interface. Na prática, essas modificações podem ser realizadas em alguns minutos pelo programador da aplicação.

3.2 Descrição da Interface

A interface de BDDmeter consiste de uma janela principal (ver figura 3), a partir da qual pode-se ativar três outras interfaces: a interface composta de

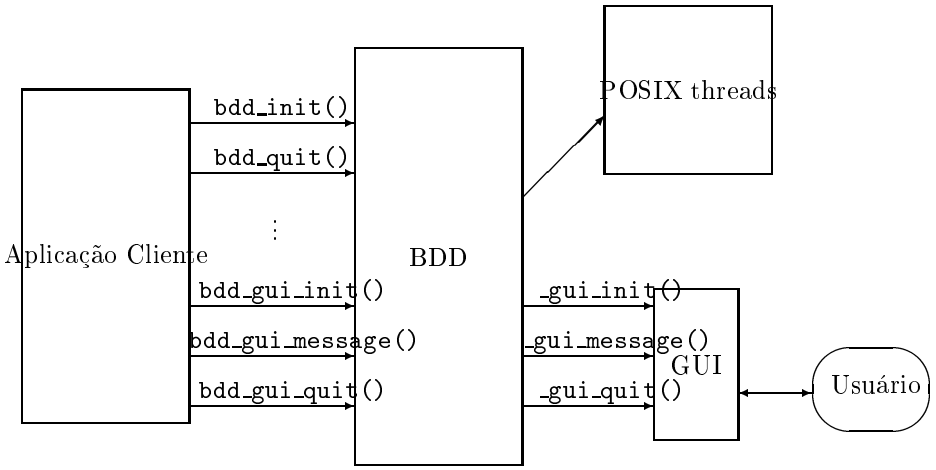

Figura 2: Padrão de arquitetura de aplicação utilizando BDDmeter. A biblioteca de BDDs executa as funções comuns e repassa as funções relacionadas com a interface para a biblioteca gráfica.

histogramas que representam o número de nós associados a cada variável (ver figura 4); o gráfico representativo da variação total do tamanho do BDD com relação ao tempo; e uma terceira janela, (ver figura 5) onde são mostradas várias estatísticas referentes ao BDD.

O usuário sempre terá aberta a janela principal, podendo escolher, a partir de botões, quais outras janelas deseja manter também ativas ou não. Os dados mostrados por cada janela de BDDmeter estarão sendo constantemente atualizados, sendo dada a possibilidade ao usuário de escolher qual intervalo de tempo deseja utilizar entre cada atualização dos dados.

Na janela referente ao número de nós de cada variável, o usuário pode escolher, através de menu *pop-up* como deseja visualizar o tamanho de cada variável, podendo decidir entre ver o tamanho individual (histograma) de cada variável ou o tamanho total associado até aquela variável (histograma acumulativo). Existe, na janela principal, um campo destinado a mostrar possíveis mensagens que a aplicação cliente possa enviar durante a execução do programa; essa funcionalidade torna-se necessária na medida em que é geralmente preciso dar ao usuário informações sobre a execução em curso.

Enquanto uma técnica de re-ordenação dinâmica de variáveis está sendo utilizada, o número de nós associado a cada uma delas varia grandemente. O usuário da biblioteca de BDDs muitas vezes não consegue discernir quais trocas de variáveis aumentam o tamanho dos BDDs e quais o reduzem, tornando assim complexo determinar os gargalos na execução devidos à má posição de algumas variáveis na ordem dos BDDs. Ao fazer uso de nossa ferramenta, torna-se mais fácil obter uma boa ordem para as variáveis, visto que é possível visualizar em

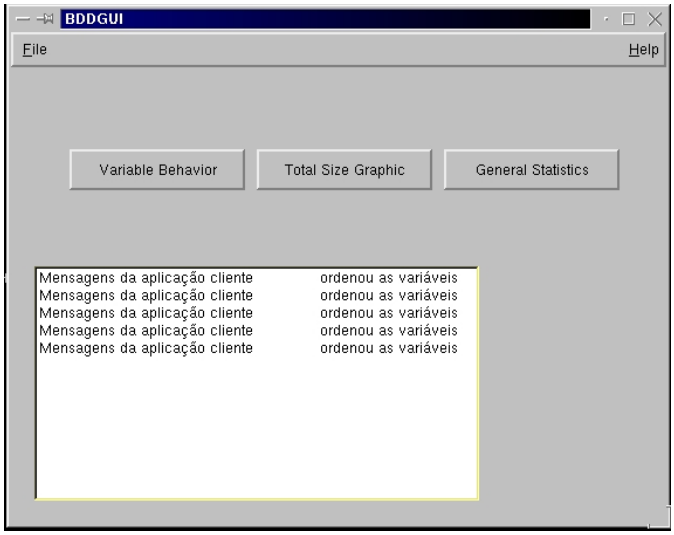

Figura 3: Janela Principal

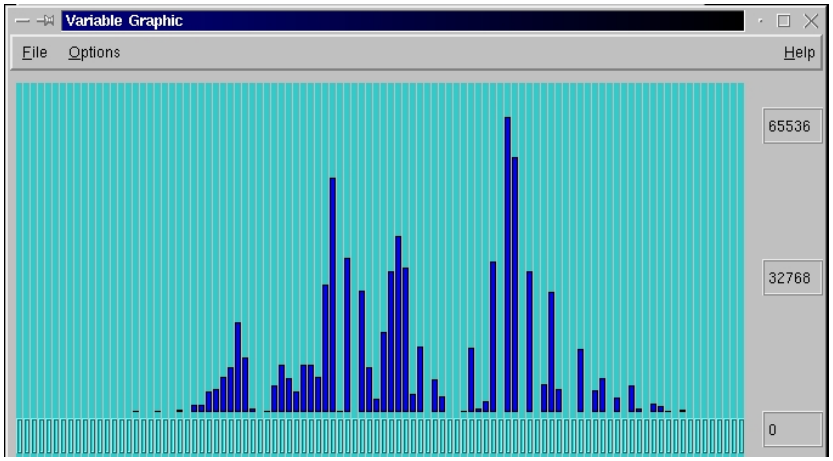

Figura 4: Gráfico das Variáveis

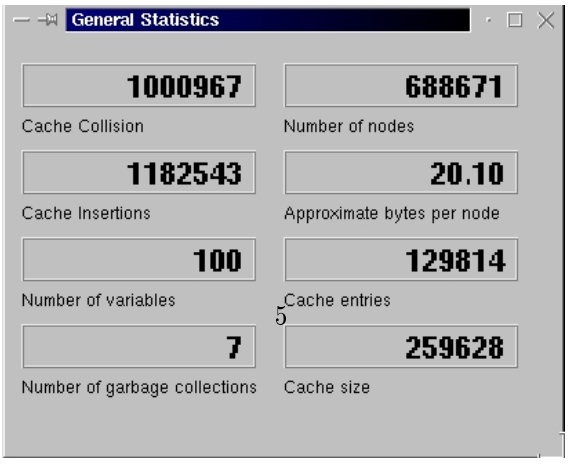

Figura 5: Estatísticas Gerais

tempo real qual o efeito que uma determinada troca de variáveis produz no tamanho do BDD, permitindo assim chegar a uma representação de tamanho satisfatível com mais rapidez.

4 Conclusões

A ferramenta ainda está em fase de desenvolvimento, mas protótipos iniciais foram apresentados a pesquisadores envolvidos no desenvolvimento e na utilização de ferramentas de verificação baseadas em BDDs. Há indícios positivos que a idéia de permitir uma visualização gráfica dinâmica de BDDs torne-se de grande utilidade para facilitar o uso de ferramentas de verificação. BDDmeter tem-se mostrado de fácil uso, tendo sido rapidamente incorporado a uma ferramenta de verificação formal, conforme o nosso objetivo inicial.

A medida que BDDmeter vai sendo apresentada a pessoas que fazem uso de BDDs, vão surgindo novas sugestões de funcionalidades para a ferramenta, portanto ela está continuamente expandindo-se e aperfeiçoando-se, tentando suprir as necessidades de possíveis usuários da ferramenta. Para aumentar a sua difusão, BDDmeter deve também ser portado em outras bibliotecas de BDDs.

Finalmente, temos também outros planos de ferramenta de apoio ao uso de BDDs, tais como o projeto BDDtuner, que possibilitará alterar dinamicamente a configuração de um dado BDD, possibilitando ao usuário reordenar as variáveis em um dado momento da execução.

Referências

[ATB94] A. Aziz, S. Taşiran, and R.K. Brayton. Bdd variable ordering for interacting finite state machines. In *31st annual conference on Design Automation conference: DAC'94*, pages 283–288, 1994.

[BRB90] K. Brace, R. Rudell, and R. Bryant. Efficient implementation of a bdd package. In *27th ACM/IEEEE Design Automation Conference*, pages 40–45, June 1990.

[Bry86] R.E. Bryant. Graph-based algorithm for boolean function manipulation. *IEEE Transactions Computers*, C(35):1035–1044, 1986.

[KF98] G. Kamhi and L. Fix. Adaptative variable ordering for symbolic model checking. In *1998 IEEE/ACM international conference on Computer-aided design: ICCAD'98*, pages 359–365. 1998.

[Rud93] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *1993 IEEE/ACM international conference on Computer-aided design: ICCAD'93*, pages 42–47. 1993.

[Sun96] Sun Microsystems. *Programming with threads*, 1996.

[Tro00] Trolltech. *QT Reference Documentation (version 2.2.1)*, 2000.