Estimativa de tempo de execução no pior caso por análise estática de código utilizando o BOUND-T

Arliones Hoeller Jr.

[1]Departamento de Automação e Sistemas (DAS)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil
arliones@das.ufsc.br

Abstract. *This paper presents a study of the* BOUND-T *tool.* BOUND-T *is a tool used to perform static analysis of code for determining worst-case execution time (WCET) of programs. This study was performed in the context of a course on real-time systems within the Automation and Systems Engineering Graduate Program at the Federal University of Santa Catarina, where students proposed a set of programs to be analyzed by several WCET tools. The paper briefly describes the* BOUND-T *tool, evaluates its performance when analyzing a benchmark for real-time applications, and presents the results obtained when analyzing the proposed programs. We present the structure of annotations needed to make most of these analysis possible and identify a few limitations on the tool.*

Resumo. *Este artigo apresenta um estudo da ferramenta* BOUND-T. *O* BOUND-T *é uma ferramenta de análise estática de código para determinar o tempo de execução no pior-caso (WCET) de programas. O estudo está contextualizado numa disciplina de sistemas de tempo-real do Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina onde os alunos propuseram um conjunto de programas-desafio a serem analisados por diferentes ferramentas de análise de WCET. Este trabalho descreve a ferramenta* BOUND-T*, avalia seu desempenho frente a um benchmark para aplicações de tempo-real, e apresenta os resultados obtidos por esta ferramenta ao analisar os desafios propostos. São apresentadas estruturas de anotações realizadas para viabilizar a análise dos programas, bem como limitações encontradas neste processo.*

1. Introdução

Tradicionalmente, o desempenho de tarefas de tempo-real é verificada por medições do tempo de execução de um conjunto de casos de testes. Definir os cenários de execução de pior caso é frequentemente difícil, e a definição de um conjunto de testes realísticos é uma tarefa pesada [Wilhelm and et. al. 2008]. Neste cenário, as ferramentas de análise estática, como o BOUND-T [Tidorum Ltd. 2011a], eliminam a necessidade deste tipo de configuração para os testes.

O BOUND-T é uma ferramenta de análise estática de código de máquina desenvolvido pela Tidorum Ltd. Através do BOUND-T é possível computar limites superiores

Este trabalho foi parcialmente financiado pela CAPES (Arliones Hoeller Jr., projeto RH-TVD 006/2008).

(*upper bounds*) de tempo de execução no pior caso (WCET - *Worst-Case Execution Time*) e de uso de pilha de tarefas de tempo-real. Além destes limites, a ferramenta permite extrair grafos de controle de fluxo (CFG) e árvores de chamadas dos programas analisados. Neste trabalho, apresentaremos um estudo de caso do emprego da ferramenta BOUND-T na análise de tarefas de tempo-real.

A Figura 1 apresenta uma visão geral do funcionamento do BOUND-T. Como uma ferramenta que realiza analise de código binário, o principal dado de entrada do BOUND-T é um arquivo binário contendo código de máquina, totalmente ligado (*linked*), em formato ELF. Informações acerca do código que gerou este binário são extraídos pelo BOUND-T a partir de informações de depuração presentes da imagem ELF. Adicionalmente, pode ser dado como entrada ao BOUND-T um arquivo de anotações contendo dados complementares para viabilizar a análise de trechos de código para os quais faltam informações suficientes para determinar cenários de pior caso. As saídas do BOUND-T, assim como a interface para sua execução, é em modo texto. Isto permite uma fácil integração com outras ferramentas e sistemas, podendo ter seus resultados facilmente analisados por um parser externo. No exemplo da Figura 1 estão representados alguns dos dados de saída da ferramenta: grafo de fluxo do programa analisado, grafo de chamadas de funções e WCET para o programa (`main`) e para funções individuais. Outra informação importante derivada da análise estática realizada pelo BOUND-T é a análise de profundidade de pilha. Abaixo um exemplo do resultado emitido pelo BOUND-T ao analisar um programa simples. O resultado foi um WCET de 12.680 ciclos (linha 1) e uma profundidade máxima da pilha de 44 bytes (linha 2).

```
1. Wcet:test:main.c:main:6-24:12680
2. Stack:test:main.c:main:6-24:SP:44
```

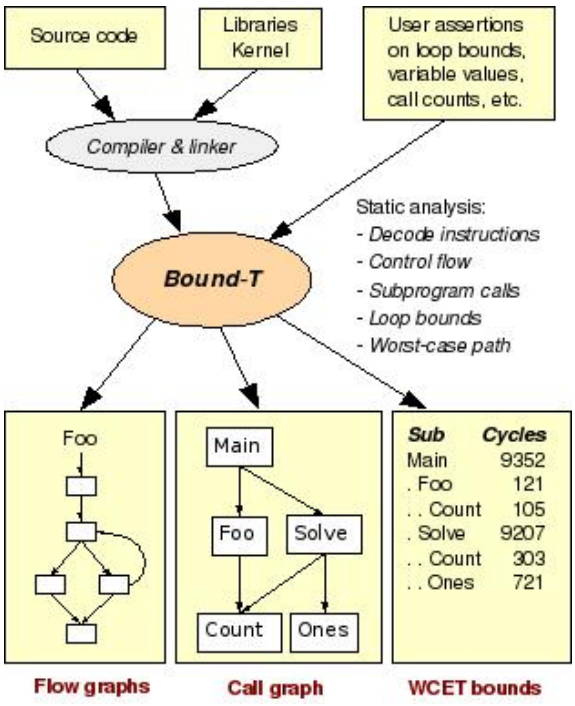

Figura 1. Visão geral do funcionamento do BOUND-T [Tidorum Ltd. 2011a].

O BOUND-T foca especialmente em processadores embarcados de pequeno e

médio porte. Ele permite modelar o comportamento de pipelines mas, atualmente, não modela o comportamento real de caches ou outros mecanismos dinâmicos, o que leva a resultados conservadores (superestimados) para tais características. Atualmente, as arquiteturas suportadas incluem a família de processadores de sinais digitais ADSP21020 da Analog Devices, ARM7-TDMI, AVR 8-bits da Atmel, 8051 da Intel, H8/300 da Renesas e Sparc V7/ERC32.

Como o problema da análise estática da execução de um programa é, em geral, indecidível, os programas submetidos à análise pelo BOUND-T não podem ter recursão e precisam ter uma estrutura de laços reduzível. O BOUND-T emprega métodos sofisticados de análise de fluxo de dados e modelagem aritmética para encontrar as variáveis vinculadas aos contadores de laços, definindo os limites de valores iniciais e finais, permitindo determinar o número máximo de vezes que um laço baseado em contador poderá ser repetido.

O BOUND-T analisa o código executável, logo, ele é independente da linguagem de programação. Ele é capaz de tratar, inclusive, programas que utilizam mais de uma linguagem. Em alguns casos, o BOUND-T pode ser sensível ao compilador utilizado. Alguns compiladores usam bibliotecas com sequências de chamada de funções que fogem ao padrão, ou "idiomas" especiais para estruturas como switch/case. O BOUND-T é, geralmente, compatível com o compilador mais utilizado para cada plataforma que suporta.

A análise de código-fonte pode ser utilizada pelo BOUND-T para facilitar a especificação dos trechos do programa que se pretende utilizar, mas não é sempre necessário. Para o Gnu GCC, por exemplo, o BOUND-T pode extrair as informações relativas ao código-fonte do binário ligado em formato ELF contendo informações de depuração.

Eventos raros, como falhas de hardware, podem ser excluídos do cenário de pior caso através de anotações feitas pelo usuário. Estas anotações podem definir previamente valores de variáveis, limites de laços, e o número de vezes que uma chamada específica é executada. Estas anotações ainda suportam a análise de programas complexos para os quais a análise estática automática não é possível. As anotações são feitas em arquivos de texto em separado. Modificações nestas anotações permitem a realização da análise em cenários diferentes permitindo, por exemplo, encontrar os WCET de um programa para diferentes entradas.

Este artigo foca em identificar cenários em que a análise automática do BOUND-T não consegue determinar o WCET de aplicações automaticamente e analisar o conjunto de anotações capazes de resolver estes casos indeterminados. Para isso, primeiramente analisamos um exemplo constante da documentação da ferramenta. Posteriormente, classificamos e selecionamos aplicações integrantes do *benchmark* de WCET do MRTC (*Mälardalen Real-Time research Centre*) [Gustafsson et al. 2010]. Finalmente, avaliamos o desempenho do BOUND-T ao analisar desafios propostos por colegas da disciplina de tempo-real do Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina. Todos os testes neste artigo foram realizados considerando um processador ARM7-TDMI com cache de instruções apenas. O compilador utilizado foi o Gnu GCC 4.4.4 [GNU 2011].

2. Análise do programa exemplo

Inicialmente, analisamos em detalhes um programa disponibilizado como exemplo na documentação do BOUND-T pelo qual é possível observar características e limitações da ferramenta para o ARM7. O exemplo consiste de dois arquivos `C`, `main.c` e `routines.c`. Há também três *headers* (*.h) e um arquivo em linguagem de montagem do ARM7, `Startup.s`. O código destes arquivos estão disponíveis no site da Tidorum [Tidorum Ltd. 2011b] [1]. O programa analisado é composto por diversas chamadas de função, sendo o grafo destas chamadas o apresentado na Figura 2.

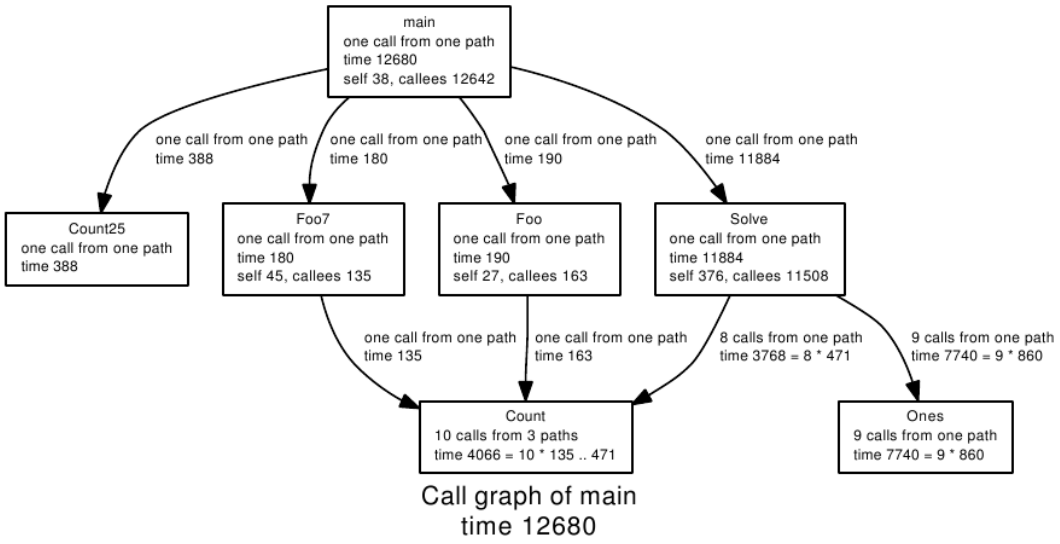

Figura 2. Grafo de chamadas do programa exemplo.

Alguns pontos neste exemplo devem ser destacados por serem importantes no processo de análise estática do programa:

- Na função `Count` há um laço cujo número de iterações depende do valor de um dos parâmetros da função (`u`);
- Na função `Ones` há um laço cujo número de iterações depende da posição do bit não nulo mais significante dentro de um dos seus parâmetros (`x`). A função conta o número de bits não nulos em `x`;
- Na função `Solve` há um laço cujo número de iterações é limitado por uma constante (`8`). Contudo, o laço pode ser terminado mais cedo por um `break`.

Para compilar e ligar o programa, a Tidorum utilizou o Keil/ARM RealView (armcc) em ambiente Windows. Como nós utilizamos o Gnu GCC, no Linux, foi necessário remover o arquivo `Startup.s`, que estava fora do padrão do nosso compilador/montador. Para substituí-lo, foi utilizado código de inicialização disponibilizado pelo próprio GCC.

[1] Por limitação no número de páginas, listagens de código-fonte foram removidos deste documento, porém estão disponíveis através da Internet.

```
subprogram "Ones"
    loop repeats 32 times; end loop;
end "Ones";

subprogram "Solve"
    loop that calls "Count"
        variable "Solve|k" <= 32;
    end loop;
end "Solve";

subprogram "main"
    loop repeats 1 times; end loop;
end "main";
```

Figura 3. Anotações do BOUND-T para o programa exemplo.

O BOUND-T não conseguiu determinar um limite de tempo para o programa automaticamente, precisando para isso de anotações sobre o comportamento de alguns pontos do programa. Estas anotações estão na Figura 3, e são comentadas abaixo:

- Na função `Ones`, uma anotação é necessária para determinar o número de repetições do laço `while(x)` que repete uma vez para cada bit não nulo de `x`. Como `x` é um valor de 32-bits, o laço repete, no máximo, 32 vezes;
- Na função `Solve`, uma anotação é necessária para determinar o número de repetições do laço `for`. Como `k` é o número de bits '1' em `*x`, que é um valor de 32 bits, `k` é no máximo 32. Este valor limita o parâmetro `u` da chamada de `Count`, e define um limite para o laço em `Count`;
- Na função `main` há um laço eterno para bloquear o programa ao final. Uma anotação é necessária para evitar que o BOUND-T pare a análise do programa. Como neste ponto toda a computação desejada já deve ter sido feita, uma anotação é incluída para que a análise considere apenas uma iteração para este laço.

O resultado da análise do WCET deste programa é de 12.680 ciclos de máquina. Também foi possível obter a profundidade máxima da pilha do programa, que foi de 44 bytes. A saída da ferramenta para esta análise é a que foi apresentada na introdução.

3. Análise do *benchmark* WCET do MRTC

O grupo MRTC (Mälardalen Real-Time research Centre), da Universidade de Mälardalen, Suécia, desenvolveu um benchmark para avaliação de ferramentas de análise de WCET no contexto do projeto SWEET [Gustafsson et al. 2010]. Muitos dos programas deste benchmark foram utilizados no *WCET Challenge 2006*. Os programas estão disponíveis publicamente [Mälardalen Real-Time research Centre 2011].

Os programas foram compilados utilizando o Gnu GCC conforme a linha abaixo, ou seja, para o processador ARM7-TDMI, sem otimizações (-O0) e com informação de depuração (-g):

```
arm-gcc -mcpu=arm7tdmi -O0 -g -o <saída>.elf <entrada>.c
```

Em seguida foram submetidos à análise pelo BOUND-T conforme a linha abaixo, onde "main" é a função cujo WCET deseja-se estimar:

Programa	Complexidade	WCET (ciclos)
bs	4	Unbounded
bsort100	6	988181
cnt	6	Unbounded
compress	6	Unbounded
cover	1	5713
crc	4	Bound-T Cracked
duff	6	Irreductible Flow Graph
expint	3	Unbounded
fac	6	Recursion
fdct	4	10961
fibcall	1	1232
fir	6	765
insertsort	6	576554
janne_complex	3	193785
jfdctint	3	48443
lcdnum	3	Unbounded
matmult	5	Unbounded
ndes	5	Unbounded
ns	6	Unbounded
qsort-exam	9	Unbounded
recursion	5	Recursion
select	9	Unbounded
st	7	Unbounded

Tabela 1. Benchmark WCET da MRTC da Universidade de Mälardalen.

```
boundt_arm7 -arm7 <entrada>.elf main
```

A Tabela 1 mostra os resultados da análise de WCET para alguns programas do benchmark. Não foram feitas tentativas de adicionar anotações para resolver os problemas encontrados nestes benchmarks. O objetivo era avaliar até onde o BOUND-T seria capaz de ir automaticamente. Basicamente, exceto para o programa "crc", em que a ferramenta deixou de funcionar durante a análise, os problemas encontrados foram a impossibilidade de determinar limites de valores para variáveis controladoras de laços (*Unbounded*), presença de grafos de fluxo de execução irredutíveis (*Irreductible Flow Graph*) e a presença de recursão no código (*Recursion*). Para os programas que o BOUND-T conseguiu realizar a análise é apresentado o valor do WCET em ciclos de máquina.

Ainda nesta tabela, é apresentado um índice de complexidade do programa em questão. Tal índice foi montado utilizando uma soma ponderada de fatores considerados complicantes do processo de análise estática de código-fonte de modo que, quanto maior a complexidade atribuída ao programa, mais difícil é o processo de análise estática do mesmo. Como exemplo de fatores complicantes de análise estática pode-se citar a presença de recursão, ponteiros para funções ou relação com bibliotecas externas ao programa (e.g., `libc`). Os fatores empregados são os mesmos apresentados pelo MRTC na descrição dos programas propostos [Mälardalen Real-Time research Centre 2011].

No contexto em que estes testes foram realizados, a disciplina de Sistemas de Tempo-Real do Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina, diversos grupos de alunos propuseram desafios para serem analisados por diversas ferramentas de análise de WCET. Sendo assim, o problema apresentado como desafio para disciplina e analisado com sucesso pelo

```
subprogram ”sort”
    loop that contains (loop that contains loop) repeats 100 times;
    end loop;
    all 5 loops in (loop that contains (loop that contains loop)) repeat 20 times;
    end loop;
end ”sort”;

subprogram ”select”
    loop that contains (loop that contains loop) repeats 20 times;
    end loop;
    all 3 loops in (loop that contains (loop that contains loop)) repeat 20 times;
    end loop;
end ”select”;

subprogram ”fac” time 0 cycles; end ”fac”;

subprogram ”facA”
    call to ”fac” repeats 0 times; end call;
end ”facA”;

subprogram ”fac_main”
    loop repeats 6 times; end loop;
    call to ”facA” repeats 0 times; end call;
end ”fac_main”;
```

Figura 4. Anotações do BOUND-T para o programa desafio proposto.

BOUND-T é uma execução sequencial dos programas considerados mais complexos pela análise apresentada na Tabela 1, ou seja, `qsort-exam`, `select` e `fac` do benchmark apresentado acima.

Para resolver o problema proposto no BOUND-T foi necessário adicionar as anotações apresentadas na Figura 4. O subprograma “sort” apresenta uma série de laços alinhados, que podem ser facilmente expressos pela linguagem de anotações do BOUND-T conforme apresentado na figura. No caso do “sort”, anotações foram necessárias para limitar o laço principal (`loop that contains (loop that contains loop)`) e os cinco laços aninhados (`all 5 loops in (...)`). Anotações similares foram realizadas para viabilizar a análise do subprograma “select”.

Para possibilitar a análise do subprograma “fac”, contudo, outro tipo de anotação se fez necessária. O BOUND-T não resolve recursões sozinho. Para resolver as chamadas recursivas à função que calcula o fatorial (`fac`), foi necessário quebrar as chamadas em 2, conforme apresentado na Figura 5, e anotar que estas chamadas não repetem nunca. Assim, o WCET pode ser calculado manualmente ao final. Os resultados obtidos foram 5.496.330 ciclos para o subprograma `qsort`, 1.107.269 ciclos para o subprograma `select` e 30 ciclos para o subprograma `fac`, totalizando 6.603.648 ciclos para todo o desafio. O WCET real, contudo, precisa levar em conta o número de repetições da função recursiva. Para obtê-lo, rodamos o analisador apenas sobre a função da recursão, considerando o tempo de `fac` como zero:

```
int facA(int n) {
    if (n == 0) return 1;
    else return (n * fac(n -1));
}

int fac(int n) {
    return facA(n);
}
```

Figura 5. Código modificado para permitir a análise de recursão.

Desafio	WCET (ciclos)
Tabela Hash	54.272
Float bit a bit	2.870
Transformada Discreta de Fourier	–
Codec ADPCM	–

Tabela 2. Resultados gerados pelo BOUND-T para os desafios.

```
$ boundt_arm7 -arm7 -assert annotations.txt desafio facA
Wcet:desafio:desafio.c:facA:220-225:24
```

Como os subprograma `fac` calcula 6 valores de fatorial (fatoriais de 0 a 5), o tempo final é dado por $WCET = (6 \times 24) + 6.603.648 = 6.606.528$ ciclos.

4. Análise dos desafios propostos

Esta seção apresenta os resultados obtidos na análise pelo BOUND-T dos desafios propostos pelas outras equipes da disciplina de tempo-real. Cada subseção seguinte trata de um dos desafios, com uma rápida descrição de cada um, o arquivo de anotações necessário para a análise e o resultado. A Tabela 2 sumariza os resultados gerados pelo BOUND-T para o WCET dos desafios propostos. Como pode ser observado, a ferramenta não foi capaz de avaliar os programas em dois dos desafios[2].

4.1. Tabela Hash

Este desafio apresenta algoritmos para a manipulação de uma tabela hash. Uma anotação foi necessária para amarrar o laço na linha 90 do programa. Como já constava no comentário existente no código, o limite de iterações do laço está associado ao tamanho máximo de uma string, que está definido no arquivo `main.h` como 512 bytes, logo, o laço repete 512 vezes.

Outro problema encontrado que necessitou de anotações foi a existência de chamadas dinâmicas a funções, ou seja, uso de ponteiros para funções. O BOUND-T permite resolver este problema através de uma anotação específica em que se passam os valores que estes ponteiros podem assumir, ou seja, as funções que podem ser atribuídas a estes ponteiros. O arquivo de anotações gerado está na Figura 6. O WCET para a função main do desafio "Tabela Hash" foi de 54.272 ciclos.

[2]Devido às limações em número de páginas, o código-fonte dos desafios aqui apresentados foram suprimidos e disponibilizados online em: `http://www.lisha.ufsc.br/~arliones/WCET_Fontes.zip`

```
subprogram ”hash”
        loop   repeats 512 times; end loop;
end ”hash”;

subprogram ”hash_insert”
        dynamic call calls ”data_2_pre_function” or ”data_3_pre_function”
                or ”data_4_pre_function” or ”data_5_pre_function”;
        end call;
end ”hash_insert”;

subprogram ”data_4_pre_function”
        dynamic call calls ”function_1” or ”function_2” or ”function_3”;
        end call;
end ”data_4_pre_function”;
```

Figura 6. Anotações para análise de WCET do desafio Tabela Hash.

```
subprogram ”main”
        dynamic call calls ”fe”; end call;
end ”main”;

subprogram ”memcpy”
        loop on line 86 repeats 1 times; end loop;
        loop on line 96 repeats 1 times; end loop;
        loop on line 106 repeats 3 times; end loop;
end ”memcpy”;
```

Figura 7. Anotações para análise de WCET do desafio float bit a bit.

4.2. Float bit a bit

Este desafio implementa algumas operações binárias sobre uma variável de ponto flutuante (*float*) implementadas através de uma função que é invocada através de um ponteiro. Também são realizadas cópias de memória utilizando a função `memcpy` da libc. Uma anotação foi necessária para resolver a chamada dinâmica a função. Esta anotação mapeia o ponteiro `pf` a seu único valor possível: `fe`. As outras anotações foram necessárias para resolver os limites dos laços de cópia de memória dentro da função `memcpy` da `libc`. Analisando o código fonte da aplicação, o pior caso de chamada ao `memcpy` realiza a cópia de 23 bytes. As anotações da Figura 7 levam isto em consideração. O resultado da análise foi um WCET de 2.870 ciclos.

4.3. Transformada Discreta de Fourier

A Transformada de Fourrier é largamente utilizada em diversos cenários, geralmente, para decompor sinais em suas componentes de frequência e amplitude. A implementação apresentada neste desafio é uma versão da Transformada Discreta de Fourrier operando sobre variáveis inteiras (ponto fixo).

O desafio proposto faz uso intensivo da biblioteca matemática `libm` que, por sua vez, faz uso de uma série de estruturas e funções internas do GCC e da `libc`. A análise

deste código se mostrou um desafio praticamente impossível. Vale destacar que, nestes casos, abordages que utilizam técnicas de medição ao invés de análise estática de código são mais indicadas. Nestes casos estas dependências pouco influenciam nos resultados de análise [Wilhelm and et. al. 2008].

Acreditamos que, devido à complexidade dos grafos de fluxo e chamadas de função gerados para esta aplicação pela interação com a `libm` o BOUND-T não foi capaz de gerar uma análise confiável. O principal problema encontrado dentro do código das bibliotecas foi a existência de uma série de chamadas dinâmicas a funções. Outro erro que nos levou a acreditar que o BOUND-T estava operando de forma inconsistente foi ele ter alegado a existência de uma recursão na função `fft` do desafio. Analisando o código, contudo, não fomos capazes de encontrar esta recursão.

4.4. Codec ADPCM

Este desafio implementa algoritmos de codificação e decodificação do formato ADPCM (Adaptive Differential Pulse-Code Modulation). Aparentemente, um bug no BOUND-T, a análise da função main deste programa falhou. Descobrimos que, se incluirmos limites por anotações às funções `__aeabi_ui2f` e `__aeabi_i2f`, o bug não ocorria. Logo, analisamos os WCET destas funções em separado, obtendo 42 ciclos como WCET da `__aeabi_ui2f` e 40 ciclos como WCET da `__aeabi_i2f`, podendo assim incluir estes WCET como anotações (Figura 8) e proceder a análise e resolução dos demais laços cujos limites não puderam ser definidos.

A partir daí, vários laços cujos limites não puderam ser definidos surgiram para as funções matemáticas `__divsf3` e `__mulsf3`. A implementação destas funções, internas ao compilador GCC para ARM, estão em linguagem de montagem no arquivo `ieee754-sf.S`, cuja análise se mostrou uma tarefa bastante complexa. Sendo assim, visando completar a análise da aplicação, consideramos tempo de execução zero para estas funções. Para obter resultados mais precisos, contudo, uma análise mais detalhada destas funções em linguagem de montagem é necessária. Como consequência, a análise de tamanho de pilha da ferramenta deixa de funcionar adequadamente.

Os laços nas funções `factorial`, `power` e `cosine` tiveram seus limites máximos (de pior caso) definidos baseados no valor do iterador `i` utilizado na função `cosine`. Haviam também chamadas dinâmicas a funções através de ponteiros para funções. Contudo, estes ponteiros são inicializados no início da função `main`, logo, têm um único comportamento, facilmente definido por anotações. As anotações utilizadas estão na Figura 8.

Mesmo acreditando que todas as anotações estejam corretas, não foi possível estimar o WCET desta aplicação porque a ferramenta BOUND-T não finalizou a análise. Todas as demais análises terminaram em menos de 5 segundos de execução. A análise desta aplicação rodou por, aproximadamente, 1 hora até consumir toda a memória disponível na máquina (4 GB). Provavelmente isto deve estar relacionado a um bug da ferramenta.

5. Conclusão

Este trabalho apresentou as principais características e funcionalidades da ferramenta de análise de tempo de execução de pior caso (WCET) BOUND-T, da Tidorum Ltd. Uma

```
subprogram "__aeabi_ui2f" time 42 cycles; end "__aeabi_ui2f";
subprogram "__aeabi_i2f" time 40 cycles; end "__aeabi_i2f";
subprogram "__divsf3" time 0 cycles; end "__divsf3";
subprogram "__mulsf3" time 0 cycles; end "__mulsf3";

subprogram "factorial"
        loop on line 22 repeats 18 times; end loop;
end " factorial ";

subprogram "power"
        loop on line 11 repeats 18 times; end loop;
end "power";

subprogram "cosine"
        loop on line 36 repeats 10 times; end loop;
end "cosine";

subprogram "main"
        dynamic call on line 77 calls "g721_encoder"; end call;
        dynamic call on line 84 calls "g721_decoder"; end call;
end "main";
```

Figura 8. Anotações para análise de WCET do desafio Codec ADPCM.

análise geral da ferramenta foi realizada seguida da apresentação de um exemplo disponível junto à documentação da ferramenta.

No contexto da disciplina DAS9007 - Sistemas de Tempo Real do Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PGEAS) da UFSC, um desafio foi proposto em que alunos, divididos em grupos, deveriam estudar ferramentas de estimativa de WCET e propôr aplicações-desafio aos demais grupos. Cada grupo, então, deveria analisar as aplicações propostas com as ferramentas que estudaram.

Neste trabalho, foi posposta uma aplicação baseada em alguns dos programas utilizados pelo benchmark de WCET do MRTC, da Universidade de Mälardalen, Suécia. A análise dos demais desafios propostos também foi realizada utilizando o Bound-T, o que revelou uma limitação da ferramenta no que diz respeito a integração com código mais complexo presente em bibliotecas do sistema (e.g., $libm$ e $libc$). A ferramenta também apresentou alguns bugs. Alguns destes bugs foram, de algum modo, contornados. Outros problemas, contudo, inviabilizaram o uso da ferramenta para a análise de dois dos desafios propostos (Transformada Discreta de Fourrier e Codec ADPCM).

Como resultado do estudo, percebemos que as ferramentas disponíveis atualmente, de modo geral, não apresentam um nível de maturidade no qual sua aplicação a qualquer programa seja possível. Mesmo nas ferramentas comerciais mais utilizadas (categoria na qual o BOUND-T se insere), uma série de anotações precisam ser feitas para viabilizar a análise. Embora muitas destas limitações existam devido a questões teóricas relacionadas a problemas não computáveis (e.g., *The Halting Problem*), muitas limitações devem-se a bugs ou incompletudes. Em alguns casos, chegar a limites razoáveis para es-

tas anotações pode ser uma tarefa bastante difícil, o que leva, frequentemente, ao uso de margens seguras, fazendo da estimativa de WCET um valor muito além do valor real, tornando sistemas de tempo-real superdimensionados.

Como trabalho futuro, pretende-se utilizar o BOUND-T para permitir a análise de WCET de aplicações utilizando o sistema operacional EPOS [Marcondes and Fröhlich 2009]. Análises iniciais já apresentaram diversos desafios nesta tarefa, como na interação entre o sistema operacional e periféricos e na implementação de funções clássicas de sistemas operacionais como sincronizadores com espera ociosa e troca de contexto. Espera-se, através deste estudo, viabilizar uma estrutura de anotações que torne viável a análise do WCET do código do sistema operacional EPOS, viabilizando assim a análise de suas aplicações.

Agradecimentos

Os autores agradecem aos professores da disciplina de sistemas de tempo-real do Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina, Rômulo Silva de Oliveira e Carlos Barros Montez, pela orientação e valiosas discussões em sala de aula. Também agradecem ao colega Emílio Wuerges e aos demais alunos que cursaram a disciplina em 2010 pelo apoio na realização do trabalho e pelos programas-desafio utilizados nos experimentos. Agradecemos também à Tidorum Ltd. pela licença acadêmica do BOUND-T.

Referências

GNU (2011). The gnu compiler collection. Internet. http://gcc.gnu.org/.

Gustafsson, J., Betts, A., Ermedahl, A., and Lisper, B. (2010). The mälardalen wcet benchmarks - past, present and future. In *Proc. of the 10th Int. Ws. on WCET Analysis*.

Marcondes, H. and Fröhlich, A. A. (2009). Modelagem e Implementacao de Escalonadores de Tempo Real para Sistemas Embarcados. In *6th Brazilian Workshop on Operating Systems*, pages 2405–2416, Bento Gonçalves, Brazil.

Mälardalen Real-Time research Centre (2011). Wcet project benchmarks. Internet. http://www.mrtc.mdh.se/projects/wcet/benchmarks.html.

Tidorum Ltd. (2011a). Bound-t website. Internet. http://www.bound-t.com/.

Tidorum Ltd. (2011b). Example of bound-t/arm7 analysis. Internet. http://www.bound-t.com/targets/arm7/example-brochure/text.html.

Wilhelm, R. and et. al. (2008). The worst-case execution-time problemoverview of methods and survey of tools. *ACM Trans. Embed. Comput. Syst.*, 7:36:1–36:53.