Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

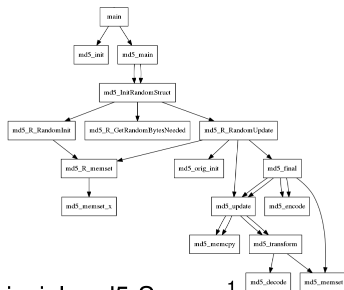
Statische Stackbedarfsanalyse

Phillip Raffeck, Tim Rheinfels, Simon Schuster, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
https://sys.cs.fau.de

Wintersemester 2022

Statische Stackbedarfsanalyse




```
1   /* Objective function */
2   max: +16 md5_orig_init +64 md5_update \
3          +64 md5_final +16 md5_memset \
4          +208 md5_transform +16 md5_encode ...;
5
6
7   /* Constraints */
8   +main = 1;
9   +md5_init +md5_main <= +main;
10  ...
```

- Beispiel: md5-Summe[1]
- Vorgehen

1. Callgraph bestimmen
2. Stackbedarf einzelner Funktionen (`gcc -fstack-usage`)
3. ILP[2] aufstellen (Nebenbedingungen aus 1., Kosten aus 2. verwenden)
4. ILP z.B. mittels `lp_solve` ⇝ **maximaler Stackbedarf**

[1] https://github.com/tacle/tacle-bench/

[2] Integer Linear Program (dt. ganzzahliges lineares Programm)

Optimierungsziel

- Jeder Stapelrahmen einer Funktion f hat eine Größe *size*
- Jede Funktion kann auf einem Pfad ein- oder mehrfach (Rekursion), insgesamt n-fach auf dem Stapel vorkommen
- Gesucht: Fluss durch den Aufrufgraphen, welcher Stapelbedarf maximiert
- Dabei müssen **Flussbedingungen** eingehalten werden
  - Aufruferbeziehung
  - Alternativen
  - ...

Optimierungsziel

$$max \sum_{\text{Funktion } f} size_f \cdot n_f$$

In `lp_solve` -Syntax:

```
max :  +64  n_f1  +48  n_f2  +42  n_f3 ;
```

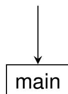
Flussbedingung: Initialer Aufruf

Semantik

Der initiale Aufruf erfolgt maximal (wahlweise auch genau) ein mal

Formalisierung

$$n_{\mathtt{main}} \leq 1$$



`lp_solve` -Syntax

```
n_main  <=  1;
```

Flussbedingung: Mehrere Vorgänger

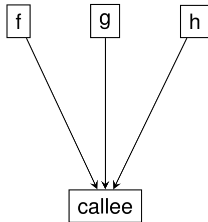
Semantik

Jede Funktion kann nur so oft ausgeführt werden, wie sie von den Vorgängern aus aufgerufen wird

Formalisierung

Sei $f_{a \to b}$ die Anzahl der Aufrufe von b durch a:

$$n_{callee} \leq \sum_{p \in \text{Aufrufer(callee)}} f_{p \to callee}$$



lp_solve -Syntax

```
n_callee  <=  + f_f_callee  + f_g_callee  + f_h_callee ;
```

Flussbedingung: Immer nur ein Nachfolger pro Funktion

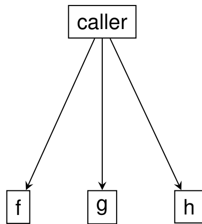
Semantik

Jede Funktionsinkarnation ruft gleichzeitig jeweils maximal eine weitere Funktion auf

Formalisierung

Sei $f_{a \to b}$ die Anzahl der Aufrufe von b durch a:

$$\sum_{c \in \text{Aufgerufene(caller)}} f_{caller \to c} \leq n_{\text{caller}}$$



lp_solve -Syntax

```
+ f_caller_f  + f_caller_g  + f_caller_h  <=  n_caller ;
```

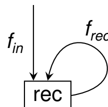
Flussbedingung: Rekursion

Semantik

Rekursive Funktionen können pro Aufruf von außen bis zu ihrer maximalen Rekursionstiefe (d) oft ausgeführt werden.

Formalisierung

$$f_{rec} \leq d_{rec} \cdot f_{in}$$
$$n_{rec} \leq f_{in} + f_{rec}$$



lp_solve -Syntax

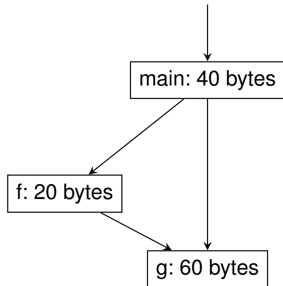
```
f_rec  <=  +42  f_in ;
n_rec  <=  f_in  +  f_rec ;
```

Beispiel

- Problemformulierung in lpsolve:

```
max: +40 n_main +20 n_f +60 n_g;

n_main <= 1;
+f_main_f +f_main_g <= n_main;
n_f <= +f_main_f;
+f_f_g <= n_f;
n_g <= +f_f_g +f_main_g;
```

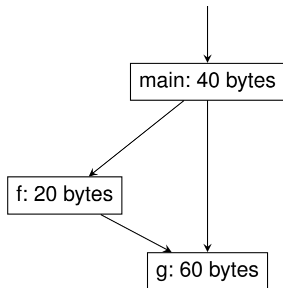


Beispiel

- Problemformulierung in lpsolve:

```
max: +40 n_main +20 n_f +60 n_g;

n_main <= 1;
+f_main_f +f_main_g <= n_main;
n_f <= +f_main_f;
+f_f_g <= n_f;
n_g <= +f_f_g +f_main_g;
```



- Ausgabe von `lp_solve`:

```
Value of objective function: 120.00000000

Actual values of the variables:
n_main                          1
n_f                             1
n_g                             1
f_main_f                        1
f_main_g                        0
f_f_g                           1
```

LP-Solve Fallstricke: Infeasible model

```
$ lp_solve infeasible.lp
This problem is infeasible
```

Infeasible Models

Logischer Widerspruch in Nebenbedingungen

Leider bietet `lp_solve` selbst direkt keine Hilfestellung zur Lokalisation. Die Entwickler empfehlen das Einführen von “slack”-Variablen:[3]

```
max: x + y;
x + 1 <= x;
y > y + 1;
x <= 20;
y <= 20;
```

```
     max: x + y
         -1000 e_1
         -1000 e_2;
x + 1 - e_1 <= x;
y + e_2 > y + 1;
x <= 20;
y <= 20;
```

```
x:    20
y:    20
e_1:   1
e_2:   1
```

[3] http://lpsolve.sourceforge.net/5.5/Infeasible.htm

LP-Solve Fallstricke: Unbounded model

```
$ lp_solve unbounded.lp
This problem is unbounded
```

Unbounded Models

Eine oder mehrere der Variablen sind nach oben unbeschränkt

Durch künstliche Beschränkung aller Variablen im System (auf einen sehr großen Wert) lassen sich unbeschränkte Variablen detektieren:

```
max: x + y + z;
z <= y + 1;
y <= 20;
```

```
max: x + y + z;
z <= y + 1;
y <= 20;
x <= 5000;
y <= 5000;
z <= 5000;
```

```
x:  5000
y:    20
z:    21
```

LP-Solve Fallstricke: Syntax

- `lp_solve` ist auf die Lösung linearer Gleichungssysteme ausgelegt
- Es ist dementsprechend nicht möglich, zwei Variablen zu multiplizieren
  - `a * b` ⇒ Syntaxfehler
  - `max: a b` ⇒ optimiert $a + b$
- Lösung in VEZS für Konstanten (Stapelrahmengrößen): C-Präprozessor:

```
#define s_main 40
#define s_f    20
#define s_g    60

max: +s_main n_main +s_f n_f +s_g n_g;
```

⇝ `stackusage / lp_solvepp`

Statische Stackbedarfsanalyse

- Statische Code-Analyse mit a[3] Tool-Suite
  1. aiT: WCET-Analyse
  2. Stack-Analyzer: Stackbedarf
  3. ...
- Installiert im CIP-Pool
- `/proj/i4ezs/tools/a3_x86/bin/a3x86`

a^3 Analyzer – Lizenzserver

Project Views Help

Home
Analyses
Setup
Information

Overview
Welcome
Recent projects

Welcome

AbsInt

License

Host ID:
License
License
Expiry d
License

Manage

Select AbsInt License Manager

Host	Port
i4alm.informatik.uni-erlangen.de (10.188.40.11)	42424

Host: i4alm.cs.fau.de Port: 42424

User: Password:

OK Cancel

ation)

er support
@absint.com
eans (online)

e in the

ail client

system.

suitable for contacting support.

Close

Filter...

Find in project Memory usage

a^3 Analyzer – Lizenzserver

Project Views Help

Home
Analyses
Setup
Information

Overview
Welcome
Recent projects

Welcome

AbsInt

License
Host ID:
License
License
Expiry d
License

Manage

Select AbsInt License Manager

Host	Port
i4alm.informatik.uni-erlangen.de (10.188.40.11)	42424

Host: i4alm.cs.fau.de Port: 42424

User: Password:

ation)
er support
@absint.com
eans (online)
e in the
ail client
Close

Zugangsdaten

Benutzer/Passwort wie bei RÜ-Helpdesk

Filter...
Find in project
Memory usage

a³ Analyzer – Neues Projekt Anlegen

Project Views Help

New	Ctrl+N
Open...	Ctrl+O
Recently opened	
Save project	
Save as project...	
Save as workspace...	
Import project	
Compare projects...	
Start all enabled analyses	F6
Start analysis	F7
Start interactive analysis	F8
Stop analysis	F4
Preferences...	Ctrl+,
Quit	Ctrl+Q

Welcome

AbsInt Advanced Analyzer for X86

...cts
» ...a project
» ...tly opened projects
» ...xamples

Documentation
» Release notes (online)
» User manual
» Safety manual
» TargetLink toolbox manual

Customer support
» support@absint.com
» Other means (online)

Info

Filter...

Find in project

Memory usage

a^3 Analyzer – Executable Angeben

Project Views Help

Home
Analyses
$f(x)$
Setup
Information

- Files
- Reporting
- Hardware
- Decoding
- Stack & value analysis
- Timing analysis
- Source files
- Configurations
- Support

Filter...

Files

Input | Output

Executables:

Browse for item

MAP file:

AIS file:

Project:

Temporary directory: /tmp/a3x86-pRqdXg

Find in project   Memory usage

a^3 Analyzer – Hardware Auswählen

Project Hardware Views Help

Home
Analyses
$f(x)$
Setup
Information

Files
Reporting
Hardware
Decoding
Stack & value analysis
Timing analysis
Source files
Configurations
Support

Filter...

Hardware

Machine Settings

General

CPU

Variant: 64-bit (Long mode)

Registers

Register contents at start of analysis

RSP:
FS.base: hex
GS.base: hex
Stack area:

Find in project
Memory usage

a^3 Analyzer – Stack-Analyse Selektieren

Project Views Help

Home

Analyses

Setup

Information

f(x) Create

f(x)

f(x) Create

*You can also use the **Symbols** or **DWARF** view to create multiple analyses of the same type by selecting the analysis entries and using the **Create analyses** action from the toolbar or context menu.*

StackAnalyzer
Stack usage analysis

ValueAnalyzer
Program value analysis

ResultCombinator
Combination of results according to formula

Control-Flow Visualizer
Visualization of control-flow graph

Filter...

Find in project

Memory usage

a³ Analyzer – Stack-Analyse Starten

Project Analysis Views Help

f(x) Create

StackAnalyzer

StackAnalyzer

ID: StackAnalyzer

Comment:

Result: n/a

Settings | Output

Configuration: Default Configuration

Dependencies: *None*

Analysis start: run

AIS file:

Expected result:

Enable ValueAnalyzer features

Home

Analyses

Setup

Information

Start all enabled analyses

Filter...

Messages | Find in project | Memory usage

a^3 Analyzer – Analyseoutput

Project Analysis Views Help

f(x) Create
StackAnalyzer
AIS files
Analysis graph

Home
Analyses
Setup
Information

StackAnalyzer

ID: StackAnalyzer
Comment:
Result: System: 96 bytes

Settings Output

Configuration: Default Configuration
Dependencies: *None*
Analysis start: run
AIS file:
Expected result:
Enable ValueAnalyzer features

Errors, warnings and info Latest log

StackAnalyzer - StackAnalyzer (0 Errors, 4 Warnings): Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second
Control-Flow & Stack Analysis
Reading binary 'stacktest'.
#1039: ELF file is not an executable, but shared object file.
#1033: ELF file is not a statically linked executable, but contains relocations.
#1034: ELF file is not a statically linked executable, but contains dynamic link information.
Using decoder for 'x86_64' and compiler 'GCC'.
Recursion 0x1125 'h' found, recursion members:
Value analyzer statistics (max-length=2, default-unroll=2, normal mode):
Loop analysis found 0 loop bounds.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 2/2 nodes * calls (non-optimizable routines: 2).
#3097: For routine 'h' the default incarnation limit of 1 is used.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 4/4 nodes * calls (non-optimizable routines: 2).
Maximum global stack height: 96
Last process took 0 s and used not more than 30 MB (RSS 13 MB) of memory
Reporting
Creating HTML report
Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second with 0 errors, 4 warnings

Filter... Messages Find in project Memory usage Overall analysis time: <1s

a^3 Analyzer – Analyseoutput

Project Analysis Views Help

Home / Analyses / Setup / Information

f(x) Create
StackAnalyzer
AIS files
Analysis graph

StackAnalyzer

ID: StackAnalyzer
Comment:
Result: System: 96 bytes

Settings | Output

Configuration: Default Configuration
Dependencies: *None*
Analysis start: run
AIS file:
Expected result:
Enable ValueAnalyzer features

Errors, warnings and info | Latest log

```
StackAnalyzer - StackAnalyzer (0 Errors, 4 Warnings): Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second
  Control-Flow & Stack Analysis
    Reading binary 'stacktest'.
    #1039: ELF file is not an executable, but shared object file.
    #1033: ELF file is not a statically linked executable, but contains relocations.
    #1034: ELF file is not a statically linked executable, but contains dynamic link information.
    Using decoder for 'x86_64' and compiler 'GCC'.
    Recursion 0x1125 'h' found, recursion members:
    Value analyzer statistics (max-length=2, default-unroll=2, normal mode):
    Loop analysis found 0 loop bounds.
    The analyzer optimized the stack graph of entry 'run' from 5/5 to 2/2 nodes *
    #3097: For routine 'h' the default incarnation limit of 1 is used.
    The analyzer optimized the stack graph of entry 'run' from 5/5 to 4/4 nodes * calls (non-optimizable routines: 2).
    Maximum global stack height: 96
    Last process took 0 s and used not more than 30 MB (RSS 13 MB) of memory
  Reporting
  Creating HTML report
  Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second with 0 errors, 4 warnings
```

$\Rightarrow$ Warnung zu ELF ignorieren

Filter... | Messages | Find in project | Memory usage | Overall analysis time: <1s

a^3 Analyzer – Callgraph

Project Analysis Views Help

f(x) Create
StackAnalyzer
AIS files
Analysis graph

Home
Analyses
Setup
Information

StackAnalyzer

ID: StackAnalyzer
Comment:
Result: System: 96 bytes

Settings | Output

Configuration: Default Configuration
Dependencies: *None*
Analysis start: run
AIS file:
Expected result:
Enable ValueAnalyzer features

Errors, warnings and info | Latest log

StackAnalyzer - StackAnalyzer (0 Errors, 4 Warnings): Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second
Control-Flow & Stack Analysis
Reading binary 'stacktest'.
#1039: ELF file is not an executable, but shared object file.
#1033: ELF file is not a statically linked executable, but contains relocations.
#1034: ELF file is not a statically linked executable, but contains dynamic link information.
Using decoder for 'x86_64' and compiler 'GCC'.
Recursion 0x1125 'h' found, recursion members:
Value analyzer statistics (max-length=2, default-unroll=2, normal mode):
Loop analysis found 0 loop bounds.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 2/2 nodes * calls (non-optimizable routines: 2).
#3097: For routine 'h' the default incarnation limit of 1 is used
The analyzer optimized the stack graph of e… …e routines: 2).
Maximum global stack height: 96
Last process took 0 s and used not more tha…
Reporting
Creating HTML report
Finished on 2020-06-15 at 17:10:14 after

Copy
Copy part
Show in call graph
Show in disassembly
Show in file
Copy path
Copy AIS annotation
Find 'limit' in DWARF
Show all folded messages of this type
Show all folded messages
Reset state of all folded messages
Clear all
Expand recursively
Collapse recursively
Expand all
Collapse all

Filter… | Messages | Find in project | Memory usage | Overall analysis time: <1s

a^3 Analyzer – Annotationstemplate kopieren

Project Analysis Graph Views Help

Recursion: <none> Exclusive Analysis graph

f(x) Create
StackAnalyzer
AIS files
Analysis graph

Maximum Stack Usage for Entry 'run': 96

run: [-8..32]
g: [-8..32]
[REC] h: [-8..32]

Toggle fold
Show address in disassembly D
Find address in DWARF Shift+D
Show source O
Copy AIS annotation
Show message Shift+M
Create analyses...
Show analysis statistics (context)
Unfold B
Unfold recursively Shift+B
Unfold routines to basic-block level Ctrl+B
Exclusive subgraph X
Scale to fit selection Z
Copy Ctrl+C
Copy address
Copy name
Go to caller C
Go to target h T
Go to neighbor
Select area around nodes Ctrl+Shift+A

Recursion bounds
Incarnation limit
Enter with
Infeasible
Not analyzed

Errors, warnings and info Latest log

StackAnalyzer - StackAnalyzer (0 Errors, 4 Warnings): Finished on 2020-06-15 at 17:10:14 a
Control-Flow & Stack Analysis
Reading binary 'stacktest'.
#1039: ELF file is not an executable, but shared object file.
#1033: ELF file is not a statically linked executable, but contains relocations.
#1034: ELF file is not a statically linked executable, but contains dynamic link informat
Using decoder for 'x86_64' and compiler 'GCC'.
Recursion 0x1125 'h' found, recursion members:
Value analyzer statistics (max-length=2, default-unroll=2, normal mode):
Loop analysis found 0 loop bounds.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 2/2 nodes * calls (non-optimizabl
#3097: For routine 'h' the default incarnation limit of 1 is used.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 4/4 nodes * calls (non-optimizabl
Maximum global stack height: 96
Last process took 0 s and used not more than 30 MB (RSS 13 MB) of memory
Reporting
Creating HTML report
Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second with 0 errors, 4 warnings

Filter... Messages Find in project Memory usage Overall analysis time: <1s

a³ Analyzer – Stack-Analyse Starten

Project Analysis Views Help

Home · Analyses · Setup · Information

f(x) Create

StackAnalyzer

ID: StackAnalyzer

Comment:

Result: n/a

Settings | Output

Configuration: Default Configuration

Dependencies: *None*

Analysis start: run

AIS file:

Expected result:

Enable ValueAnalyzer feature

Start all enabled analyses

Filter...

Messages · Find in project · Memory usage

⇒ `.ais`-Datei für benutzerdefinierte Annotationen

a^3 Analyzer – Annotationstemplate kopieren

Project Analysis Graph Views Help

Home | Analyses | Setup | Information

f(x) Create
StackAnalyzer
AIS files
Analysis graph

Recursion: <none> Exclusive | Analysis graph

Maximum Stack Usage for Entry 'run': 96

run: [-8..32]

g: [-8..32]

[REC] h: [-8..32]

Toggle fold
Show address in disassembly D
Find address in DWARF Shift+D
Show source O
Copy AIS annotation → Recursion bounds / Incarnation limit / Enter with / Infeasible / Not analyzed
Show message Shift+M
Create analyses...
Show analysis statistics (context)
Unfold B
Unfold recursively Shift+B
Unfold routines to basic-block level Ctrl+B
Exclusive subgraph X
Scale to fit selection Z
Copy Ctrl+C
Copy address
Copy name
Go to caller C
Go to target h T
Go to neighbor
Select area around nodes Ctrl+Shift+A

Errors, warnings and info | Latest log

StackAnalyzer - StackAnalyzer (0 Errors, 4 Warnings): Finished on 2020-06-15 at 17:10:14 a
Control-Flow & Stack Analysis
Reading binary 'stacktest'.
#1039: ELF file is not an executable, but shared object file.
#1033: ELF file is not a statically linked executable, but contains relocations.
#1034: ELF file is not a statically linked executable, but contains dynamic link informat
Using decoder for 'x86_64' and compiler 'GCC'.
Recursion 0x1125 'h' found, recursion members:
Value analyzer statistics (max-length=2, default-unroll=2, normal mode):
Loop analysis found 0 loop bounds.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 2/2 nodes * calls (non-optimizabl
#3097: For routine 'h' the default incarnation limit of 1 is used.
The analyzer optimized the stack graph of entry 'run' from 5/5 to 4/4 nodes * calls (non-optimizabl
Maximum global stack height: 96
Last process took 0 s and used not more than 30 MB (RSS 13 MB) of memory
Reporting
Creating HTML report
Finished on 2020-06-15 at 17:10:14 after analyzing for 1 second with 0 errors, 4 warnings

Filter... | Overall analysis time: <1s

Ais-Notationen

- Auch als C-Kommentar verwendbar
- `// ai: routine "h" recursion bound : 0 .. 42;`

a^3 Analyzer – Kommentar-Parsing Aktivieren

Project Views Help

Home
Analyses
Setup
Information

- Files
- Reporting
- Hardware
- Decoding
- Stack & value analysis
- Timing analysis
- Source files
- Configurations
- Support

Filter...

Decoding

Annotations

- [ ] Use legacy AIS annotations
- [x] Extract annotations from executables
- [x] Extract annotations from source files

AIS source code annotation prefix: ai `// ai: loop here bound: …`

Decoding

- [ ] Use only safe patterns
- [ ] Always read program headers
- [x] Enable value-iterative decoding
- [x] Enable trace-iterative decoding
- [x] Use automatic annotations for call graph creation and disassembly

DWARF Debug Information

- [x] Extract debug information
  - [ ] Extract volatile memory regions
  - [ ] Extract constant memory regions

Find in project Memory usage Overall analysis time: <1s

Aufgabenstellung

- Existierende Implementierung: Array-Datenstruktur
- Vorgegebene Funktionen: Sortieren, Maximumssuche, ...
- Aufgaben

1. Dynamische Analyse
   1.1 Thread erstellen
   1.2 Stack initialisieren
   1.3 Programm (mit Eingabedaten) ausführen
   1.4 Stackverbrauch messen
2. Statische Analyse
   2.1 ILP aus Aufrufgraph aufstellen
   2.2 Mittels `lp_solve` lösen
   2.3 Analyse mittels a^3 Stack-Analyzer