Analysis and Verification of Pointer Programs

Introduction

Winter Semester 2016/17; 20 October, 2016

C. Jansen, C. Matheja, T. Noll

Software Modeling and Verification Group

RWTH Aachen University

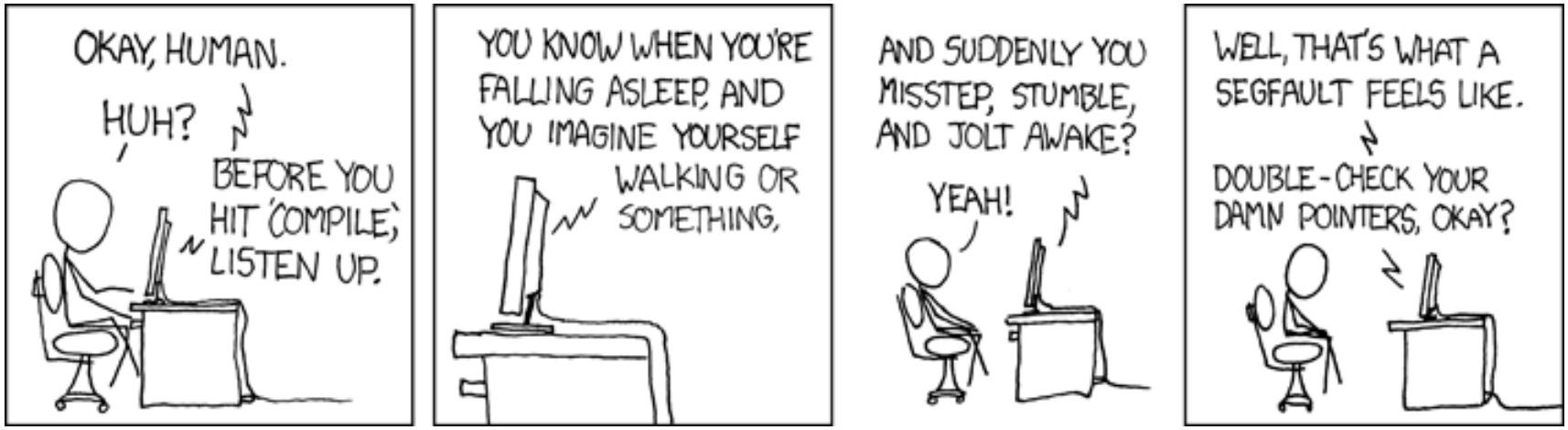
`https://moves.rwth-aachen.de/teaching/ws-1617/pointer/`

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline

Pointer-Related Software Errors



https://xkcd.com/371

Sequential programming errors

- Dereferencing invalid pointers
- Creation of memory leaks
- Invalidation of data structures

Concurrent programming errors

- Deadlocks
- Data races, ...

Problems

Analysis problem: unbounded state spaces

- Infinite data domains
- Dynamic storage (de-)allocation
- Recursive procedures
- Dynamic thread creation

Problems ... and Solutions

Analysis problem: unbounded state spaces

- Infinite data domains
- Dynamic storage (de-)allocation
- Recursive procedures
- Dynamic thread creation

Solution: abstraction

- Automata-based: regular model checking, forest automata
- Logic-based: shape analysis, separation logic
- Graph-based: graph grammars, graph transformation systems
- Extensions for concurrency

Problems

Analysis problem: unbounded state spaces

- Infinite data domains
- Dynamic storage (de-)allocation
- Recursive procedures
- Dynamic thread creation

Solution: abstraction

- Automata-based: regular model checking, forest automata
- Logic-based: shape analysis, separation logic
- Graph-based: graph grammars, graph transformation systems
- Extensions for concurrency

Outline

Aims of this Seminar

Goals

Aims of this seminar

- Independent understanding of a scientific topic
- Acquiring, reading and understanding scientific literature
- Writing of your own report on this topic
- Oral presentation of your results

Aims of this Seminar

Requirements on Report

Your report

- Independent writing of a report of $\approx$ 15 pages
- Complete set of references to all consulted literature
- Correct citation of important literature
- Plagiarism: taking text blocks (from literature or web) without source indication causes immediate exclusion from this seminar
- Font size 12pt with “standard” page layout
- Language: German or English
- We expect the correct usage of spelling and grammar
  - $\geq$ 10 errors per page $\Longrightarrow$ abortion of correction
- Report template will be made available on seminar web page

Aims of this Seminar

Requirements on Talk

Your talk

- Talk of about 45 (= 40 + 5) minutes
- Focus your talk on the audience
- Descriptive slides:
  - $\leq$ 15 lines of text
  - use (base) colors in a useful manner
- Language: German or English
- No spelling mistakes please!
- Finish in time. Overtime is bad
- Ask for questions

Final Preparations

Preparation of your talk

- Setup laptop and projector ahead of time
- Use a (laser) pointer
- Number your slides
- Multiple copies: laptop, USB, web
- Have backup slides ready for expected questions

Outline

Important Dates

Deadlines

- 21 Nov: Detailed outline of report due
- 12 Dec: Report due
- 09 Jan: Presentation slides due
- Wed 18 Jan AM/Thu 19 Jan PM: Seminar

Important Dates

Deadlines

- 21 Nov: Detailed outline of report due
- 12 Dec: Report due
- 09 Jan: Presentation slides due
- Wed 18 Jan AM/Thu 19 Jan PM: Seminar

Missing a deadline causes immediate exclusion from the seminar

Important Dates

Selecting Your Topic

Procedure

- You obtain(ed) a list of topics of this seminar.
- Classified according to BSc/MSc level.
- Indicate the preference of your topics (first, second, third).
- Return sheet by Monday (24 October) via e-mail/to secretary.
- We do our best to find an adequate topic-student assignment.
  - disclaimer: no guarantee for an optimal solution
- Assignment will be published on website by Tuesday.

Important Dates

Selecting Your Topic

Procedure

- You obtain(ed) a list of topics of this seminar.
- Classified according to BSc/MSc level.
- Indicate the preference of your topics (first, second, third).
- Return sheet by Monday (24 October) via e-mail/to secretary.
- We do our best to find an adequate topic-student assignment.
  - disclaimer: no guarantee for an optimal solution
- Assignment will be published on website by Tuesday.

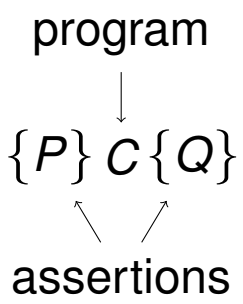
Withdrawal

- You have up to three weeks to refrain from participating in this seminar.
- Later cancellation (by you or by us) causes a not passed for this seminar and reduces your (three) possibilities by one.

Outline

Hoare Logic

Classical program verification is based on Hoare triples:

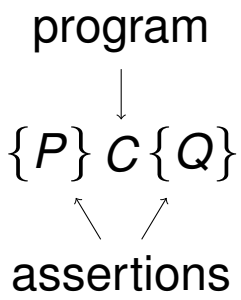


States: $\sigma : \mathit{Var} \dashrightarrow \mathbb{Z}$

$$\models \{P\}\ C\ \{Q\} \quad \Leftrightarrow \quad \forall \sigma\ .\ \sigma \models P \ \wedge\ \langle C, \sigma \rangle \rightarrow \sigma' \Rightarrow \sigma' \models Q$$

Hoare Logic

Classical program verification is based on Hoare triples:



States: $\sigma : \mathit{Var} \dashrightarrow \mathbb{Z}$

$$\models \{P\}\ C\ \{Q\} \quad \Leftrightarrow \quad \forall \sigma\ .\ \sigma \models P \ \wedge\ \langle C, \sigma \rangle \rightarrow \sigma' \Rightarrow \sigma' \models Q$$

Components

- Programming language + semantics (`WHILE` language)
- Assertion language + semantics (first-order arithmetic)
- Proof rules (not shown)

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {
  if (x != 0) {
    l := x.left;
    r := x.right;
    cleanup(x.left);
    cleanup(x.right);
    free(x);
  }
}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                                   {tree x}
  if (x != 0) {
    l := x.left;
    r := x.right;
    cleanup(x.left);
    cleanup(x.right);
    free(x);
  }
}                                              {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                                   {tree x}
  if (x != 0) {                                {∃y,z . x ↦ (y,z) * tree y * tree z}
    l := x.left;
    r := x.right;
    cleanup(x.left);
    cleanup(x.right);
    free(x);
  }
}                                              {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                                  {tree x}
  if (x != 0) {                               {∃y,z . x ↦ (y,z) * tree y * tree z}
    l := x.left;                              {∃z . x ↦ (l,z) * tree l * tree z}
    r := x.right;
    cleanup(x.left);
    cleanup(x.right);
    free(x);
  }
}                                             {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                                  {tree x}
  if (x != 0) {                               {∃y,z . x ↦ (y,z) * tree y * tree z}
    l := x.left;                              {∃z . x ↦ (l,z) * tree l * tree z}
    r := x.right;                             {x ↦ (l,r) * tree l * tree r}
    cleanup(x.left);
    cleanup(x.right);
    free(x);
  }
}                                             {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                                  {tree x}
  if (x != 0) {                               {∃y,z . x ↦ (y,z) * tree y * tree z}
    l := x.left;                              {∃z . x ↦ (l,z) * tree l * tree z}
    r := x.right;                             {x ↦ (l,r) * tree l * tree r}
    cleanup(x.left);                          {x ↦ (l,r) * emp * tree r}
    cleanup(x.right);
    free(x);
  }
}                                             {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                                  {tree x}
  if (x != 0) {                               {∃y,z . x ↦ (y,z) * tree y * tree z}
    l := x.left;                              {∃z . x ↦ (l,z) * tree l * tree z}
    r := x.right;                             {x ↦ (l,r) * tree l * tree r}
    cleanup(x.left);                          {x ↦ (l,r) * emp * tree r}
    cleanup(x.right);                         {x ↦ (l,r) * emp * emp}
    free(x);
  }
}                                             {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

```
cleanup(x) {                          {tree x}
  if (x != 0) {                       {∃y,z . x ↦ (y,z) * tree y * tree z}
    l := x.left;                      {∃z . x ↦ (l,z) * tree l * tree z}
    r := x.right;                     {x ↦ (l,r) * tree l * tree r}
    cleanup(x.left);                  {x ↦ (l,r) * emp * tree r}
    cleanup(x.right);                 {x ↦ (l,r) * emp * emp}
    free(x);                          {emp * emp * emp}
  }
}                                     {emp}
```

Separation Logic (on Trees)

States: $\sigma = (s, h)$, stack $s : \mathit{Var} \dashrightarrow \mathbb{Z}$, heap $h : \mathbb{Z} \dashrightarrow \mathbb{Z}^2$

Assertions: $x \in \mathit{Var}, a, b \in \mathit{Var} \cup \mathbb{Z}$

$$P ::= \mathit{emp} \mid x \mapsto (a, b) \mid P * P \mid \mathrm{tree}\ x \mid a < b \mid \exists x\ .\ P \mid \neg P \mid P \vee P$$

$$\mathrm{tree}\ x \triangleq (\mathit{emp} \wedge x = 0) \vee (\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z)$$

`cleanup(x) {`	$\{\mathrm{tree}\ x\}$
`  if (x != 0) {`	$\{\exists y, z\ .\ x \mapsto (y, z) * \mathrm{tree}\ y * \mathrm{tree}\ z\}$
`    l := x.left;`	$\{\exists z\ .\ x \mapsto (l, z) * \mathrm{tree}\ l * \mathrm{tree}\ z\}$
`    r := x.right;`	$\{x \mapsto (l, r) * \mathrm{tree}\ l * \mathrm{tree}\ r\}$
`    cleanup(x.left);`	$\{x \mapsto (l, r) * \mathit{emp} * \mathrm{tree}\ r\}$
`    cleanup(x.right);`	$\{x \mapsto (l, r) * \mathit{emp} * \mathit{emp}\}$
`    free(x);`	$\{\mathit{emp} * \mathit{emp} * \mathit{emp}\}$
`  }`	$\{\mathit{emp}\}$
`}`	$\{\mathit{emp}\}$

Local Reasoning

The proof relies on a frame property:

$$\frac{\{\text{tree } l\}\, \mathit{cleanup}(l)\, \{\mathit{emp}\}}{\{\text{tree } l * \text{tree } r\}\, \mathit{cleanup}(l)\, \{\mathit{emp} * \text{tree } r\}}$$

Local Reasoning

The proof relies on a frame property:

$$\frac{\{\text{tree } l\}\, \mathit{cleanup}(l)\, \{\mathit{emp}\}}{\{\text{tree } l * \text{tree } r\}\, \mathit{cleanup}(l)\, \{\mathit{emp} * \text{tree } r\}}$$

General frame rule:

$$[\mathit{frame}]\frac{\{P\}\ C\ \{Q\}}{\{P * R\}\ C\ \{Q * R\}} \text{ if } \mathit{Mod}(C) \cap \mathit{FV}(R) = \emptyset$$

Local Reasoning

The proof relies on a frame property:

$$\frac{\{\text{tree } l\}\, cleanup(l)\, \{emp\}}{\{\text{tree } l * \text{tree } r\}\, cleanup(l)\, \{emp * \text{tree } r\}}$$

General frame rule:

$$[frame]\frac{\{P\}\ C\ \{Q\}}{\{P * R\}\ C\ \{Q * R\}} \text{ if } Mod(C) \cap FV(R) = \emptyset$$

But

$$\frac{\{x \mapsto (1,2)\}\ x.left := 2\ \{x \mapsto (2,2)\}}{\{x \mapsto (1,2) \wedge y \mapsto (1,2)\}\ x.left := 2\ \{x \mapsto (2,2) \wedge y \mapsto (1,2)\}}$$

is not valid if $x = y$.

Outline

1. B: Local Shape Analysis [Noll]

- Shape analysis: discover shapes of data structures maintained on the heap
- Addresses more expressive properties than “standard” pointer analysis
  - not only “can x and y be aliases?”
  - but also “does z point to an acyclic linked list?”
- Here specified by separation logic formulae (“symbolic heaps”), e.g., for list segments:

$$\mathsf{ls}(x, y) \triangleq x \neq y \wedge (x \mapsto y \vee (\exists z. x \mapsto z * \mathsf{ls}(z, y)))$$

- Compute abstract state space of given program by means of symbolic execution, e.g., for dequeuing:

$$\{\mathsf{ls}(x, y)\}\ v := x; x := x.\mathit{next}\ \{v \mapsto x \wedge (x = y \vee \mathsf{ls}(x, y))\}$$

- Ensure termination of state-space exploration by widening

2. B: Compositional Shape Analysis [Matheja]

What is it all about?

- Shape analysis: Discover data structures encountered during a program's execution
- Compositional analysis: Every procedure is analyzed independently of its callers
- Abduction problem: Identify missing part ? in a formula to make $\varphi * ? \rightarrow \psi$ valid
- Use abduction to generate pre/post conditions for each procedure individually

Disadvantages of Whole-Program Analysis

- Limited scalability ($<$ 1000 LOC)
- Not applicable to incomplete programs

Advantages of Compositional Analysis

- More akin to human thinking
- Efficient usage of multi-core computers
- Scales up to industrial code bases: Linux kernel (2.5 MLOC) analyzed in $\sim$ 1750 seconds

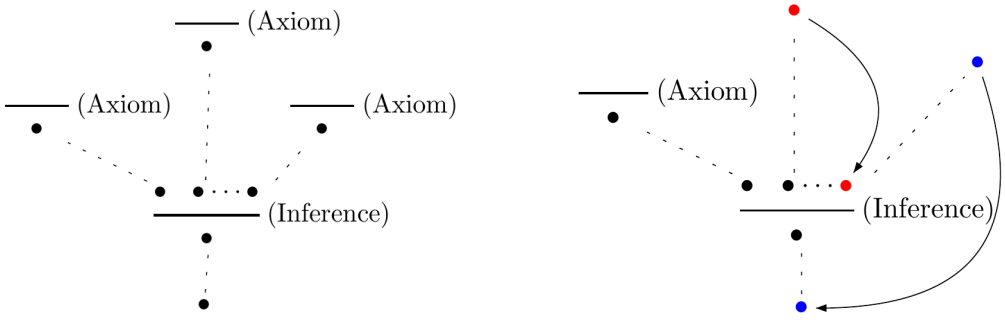
3. B: Cyclic Entailment Proofs [Matheja]

What is it all about?

- Entailment problem: $\varphi \rightarrow \psi$ valid?
- Needed to discharge rule of consequence
- Essential for automated verification based on Separation Logic

Main Ideas

- Entailments are usually discharged by constructing proof trees
- Inductive reasoning often leads to infinite proof trees
- Find cycles in these trees to automate inductive reasoning



4. B/M: Arithmetic Strengthening for Shape Analysis [Matheja]

What is it all about?

- Numerical properties: ”x is always greater than 0 at program location 23”
- Shape properties: ”$\mathtt{p}$ points to the head of a list at location 42”
- Real world programs use arithmetic and pointers together

Example

```
while(*i < n) { t = (List*) malloc(sizeof(List));
                t->next = curr;
                t->data = f(t->data);
                curr = t;    // curr = null?
                *i = *i + 1; }
```

Main Ideas

- Improve precision of shape analysis by detecting spurious counterexamples
- Apply numerical analysis refine programs and rerun shape analysis

5. B/M: Amortised Resource Analysis [Matheja]

Example

```
for(ptr = start; ptr != null; ptr = ptr.next) {
  expensiveOperation( ptr.data );
  ptr = ptr.next; }
```

What is it all about?

- What is the complexity of the program from above?
- Resource usage depends on length of list
- Handled nicely by armortised resource analysis
- Use Separation Logic to automatically derive complexity bounds

Main Ideas

- Combine Separation Logic with resources
- $\{R\}\texttt{consume}(R)\{\texttt{emp}\}$: ”consume R at a given cost”
- Use a type system for automated amortized complexity analysis

6. M: Concurrent Separation Logic [Noll]

$$\frac{\{P_1\}C_1\{Q_1\} \quad \{P_2\}C_2\{Q_2\}}{\{P_1 * P_1\}C_1 \parallel C_2\{Q_1 * Q_2\}}$$

Concurrent Separation Logic (CSL)

- Extension of SL that allows modular reasoning about threads that access separate or common storage
- Challenge: prove safety of access to common storage where mutual exclusion not guaranteed
- Solved by concept of ownership of resources and transfer of ownership
- Show: at any time, storage can be partitioned according to ownership such that safe access is guaranteed

2016 Gödel Prize

The 2016 Gödel Prize was awarded by EATCS to Stephen Brookes and Peter W. O'Hearn for their invention of Concurrent Separation Logic.

Outline

Some Final Hints

Hints

- Take your time to understand your literature.
- Be proactive! Look for additional literature and information.
- Discuss the content of your report with other students.
- Be proactive! Contact your supervisor on time.
- Prepare the meeting(s) with your supervisor.
- Forget the idea that you can prepare a talk in a day or two.

Some Final Hints

Hints

- Take your time to understand your literature.
- Be proactive! Look for additional literature and information.
- Discuss the content of your report with other students.
- Be proactive! Contact your supervisor on time.
- Prepare the meeting(s) with your supervisor.
- Forget the idea that you can prepare a talk in a day or two.

We wish you success and look forward to an enjoyable and high-quality seminar!