Locally Abstract, Globally Concrete Semantics

TECHNISCHE UNIVERSITÄT DARMSTADT

A Painless Introduction By Dr. Hähnle

Reiner Hähnle
Software Engineering Group

Software Engineering Group

Wanted!

A Trace Semantics for Concurrent Programs

Input: a concurrent program *P*, an initial state σ

Output: all traces τ of *P* starting in σ

Wanted!

TECHNISCHE UNIVERSITÄT DARMSTADT

A Trace Semantics for Concurrent Programs

Input: a concurrent program P, an initial state σ

Output: all traces τ of P starting in σ

Design goals

- Suitable as a semantics for a program verifier
- Modular in the sense of:
  - can add new language concepts incrementally
  - can evaluate locally one statement at a time
  - can constructively and incrementally generate finite/initial traces

Software Engineering Group

Existing Solutions

SOS	Plotkin et al.	not modular
Transition traces	Brookes	very complex, infinite objects
Action traces	Brookes	complex, limited coverage
Hybrid traces w/ explicit heap	Kamburjan	complex, infinite objects

The Challenge

m(x) { body } *// Executing asynchronously called method*

The Challenge

TECHNISCHE UNIVERSITÄT DARMSTADT

m(x) { body } `// Executing asynchronously called method`

We don't know the context needed for local evaluation

- Caller ?
- Processor (executing object) ?
- Parameter values ?
- Destiny ?

The Challenge

```
m(x) { body } // Executing asynchronously called method
```

We don't know the context needed for local evaluation

- Caller ?
- Processor (executing object) ?
- Parameter values ?
- Destiny ?

```
s0 ; si // Composite statement
```

How to evaluate the s_i incrementally?

The Approach (I)

Unknown context represented with symbolic expressions

The Approach (I)

TECHNISCHE UNIVERSITÄT DARMSTADT

Unknown context represented with symbolic expressions

Parametric Execution Traces

Symbolic state: Partial mapping from variables to symbolic expressions

The Approach (I)

TECHNISCHE UNIVERSITÄT DARMSTADT

Unknown context represented with symbolic expressions

Parametric Execution Traces

Symbolic state: Partial mapping from variables to symbolic expressions

Variable	Value	
x	X	value expression is symbolic variable (caps)
X	$*$	symbolic variables have unknown value
Y	X	no chains
z	$x+1+1$	maximally evaluated

The Approach (I)

Unknown context represented with symbolic expressions

Parametric Execution Traces

Symbolic state: Partial mapping from variables to symbolic expressions

Variable	Value
x	X
X	$*$
Y	$*$
z	$X + 2$

value expression is symbolic variable (caps)
symbolic variables have unknown value
no chains
maximally evaluated

The Approach (I)

Unknown context represented with symbolic expressions

Parametric Execution Traces

Symbolic state: Partial mapping from variables to symbolic expressions

Variable	Value
x	X
X	$*$
Y	
z	$X+2$

value expression is symbolic variable (caps)
symbolic variables have unknown value
no chains
maximally evaluated

Traces over symbolic states with path conditions

The Approach (I)

TECHNISCHE UNIVERSITÄT DARMSTADT

Unknown context represented with symbolic expressions

Parametric Execution Traces

Symbolic state: Partial mapping from variables to symbolic expressions

Variable	Value
x	X
X	$*$
Y	
z	$X+2$

value expression is symbolic variable (caps)
symbolic variables have unknown value
no chains
maximally evaluated

Traces over symbolic states with path conditions

$$pc_0 \triangleright \tau_0 \quad = \quad \{Y_0 > 0\} \triangleright \langle [x_0 \mapsto Y_0 + 42,\ Y_0 \mapsto *] \rangle \curvearrowright [x_0 \mapsto 17,\ Y_0 \mapsto *]$$

Software Engineering Group

The Approach (II)

TECHNISCHE UNIVERSITÄT DARMSTADT

How to ensure that incrementally produced traces are well-formed?

For example: a method must have been called before its body is executed

Software Engineering Group

The Approach (II)

TECHNISCHE UNIVERSITÄT DARMSTADT

How to ensure that incrementally produced traces are well-formed?

For example: a method must have been called before its body is executed

Events

Order-sensitive concurrent statements inject events into traces

$\tau \curvearrowright \mathit{invEv}(\mathit{mtdName}, \mathit{value})$

Software Engineering Group

The Approach (II)

How to ensure that incrementally produced traces are well-formed?

For example: a method must have been called before its body is executed

Events

Order-sensitive concurrent statements inject events into traces

$\tau \curvearrowright \mathit{invEv}(\mathit{mtdName}, \mathit{value})$

Well-Formedness

$\mathit{wf}(\tau \curvearrowright \mathit{invREv}(m, v)) = \mathit{wf}(\tau) \land \#_\tau(\mathit{invEv}(m, v)) > \#_\tau(\mathit{invREv}(m, v))$

The Approach (III)

TECHNISCHE UNIVERSITÄT DARMSTADT

How to evaluate composite statements incrementally?

Continuations

Traces end with a continuation marker containing a statement:

$$pc \triangleright \tau \cdot \lambda(s) \in \mathbf{CTr}$$

Meaning: traces of s must still be generated and appended to τ

Software Engineering Group

How to Design an LAGC Semantics in 4 Easy Steps?

1. For each statement in the language design a local evaluation rule:

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

How to Design an LAGC Semantics in 4 Easy Steps?

TECHNISCHE UNIVERSITÄT DARMSTADT

1. For each statement in the language design a local evaluation rule:

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

2. Design a trace composition rule extending traces and continuations:
   Given concrete trace *sh* and continuation $\lambda(s)$:

Software Engineering Group

How to Design an LAGC Semantics in 4 Easy Steps?

TECHNISCHE UNIVERSITÄT DARMSTADT

1. For each statement in the language design a local evaluation rule:

$$\mathrm{val}_\sigma : \mathit{Stmt} \rightarrow 2^{\mathbf{CTr}}$$

2. Design a trace composition rule extending traces and continuations:
   Given concrete trace sh and continuation $\lambda(s)$:
   - Evaluate first statement of s to τ in state $\mathrm{last}(sh)$, with $\lambda(s')$ remaining

Software Engineering Group

How to Design an LAGC Semantics in 4 Easy Steps?

TECHNISCHE UNIVERSITÄT DARMSTADT

1. For each statement in the language design a local evaluation rule:

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

2. Design a trace composition rule extending traces and continuations:
   Given concrete trace sh and continuation $\lambda(s)$:
   - Evaluate first statement of s to τ in state $\mathrm{last}(sh)$, with $\lambda(s')$ remaining
   - Stitch sh and τ together, ensure well-formedness, continuation now $\lambda(s')$

Software Engineering Group

How to Design an LAGC Semantics in 4 Easy Steps?

TECHNISCHE UNIVERSITÄT DARMSTADT

1. For each statement in the language design a local evaluation rule:

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

2. Design a trace composition rule extending traces and continuations: Given concrete trace sh and continuation $\lambda(s)$:
   - Evaluate first statement of s to τ in state $\mathrm{last}(sh)$, with $\lambda(s')$ remaining
   - Stitch sh and τ together, ensure well-formedness, continuation now $\lambda(s')$

3. Design well-formedness predicate ensuring proper event order

Software Engineering Group

How to Design an LAGC Semantics in 4 Easy Steps?

TECHNISCHE UNIVERSITÄT DARMSTADT

1. For each statement in the language design a local evaluation rule:

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

2. Design a trace composition rule extending traces and continuations:
   Given concrete trace *sh* and continuation $\lambda(s)$:
   - Evaluate first statement of s to τ in state $\mathrm{last}(sh)$, with $\lambda(s')$ remaining
   - Stitch *sh* and τ together, ensure well-formedness, continuation now $\lambda(s')$

3. Design well-formedness predicate ensuring proper event order

4. To obtain global trace semantics of program P:

   Exhaustively apply (2.) starting with $\langle \sigma_{Init} \rangle$ and continuation $\lambda(P)$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

$$\mathrm{val}_\sigma(\texttt{skip}) \quad = \quad \{\emptyset \triangleright \langle\sigma\rangle \cdot \lambda()\}$$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

$$\mathrm{val}_\sigma(x := e) \quad = \quad \{\emptyset \triangleright \langle\sigma\rangle \curvearrowright \sigma[x \mapsto \mathrm{val}_\sigma(e)] \cdot \lambda(\lozenge)\}$$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

$$\mathrm{val}_\sigma(\texttt{if}\ \mathsf{e}\ \{\ s\ \}) \quad = \quad \{\{\ \mathrm{val}_\sigma(\mathsf{e}) = \mathtt{tt}\} \triangleright \langle\sigma\rangle \cdot \lambda(s), \qquad \}$$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

$$\mathrm{val}_\sigma(\texttt{if}\ \mathsf{e}\ \{\ \mathsf{s}\ \}) \quad = \quad \{\{\mathrm{val}_\sigma(\mathsf{e}) = \mathrm{tt}\} \triangleright \langle\sigma\rangle \cdot \lambda(\mathsf{s}),\ \ \{\mathrm{val}_\sigma(\mathsf{e}) = \mathrm{ff}\} \triangleright \langle\sigma\rangle \cdot \lambda(\text{\char"2B2E})\ \}$$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

$$\mathrm{val}_\sigma(r; s) \quad = \quad \{ \qquad\qquad\qquad\qquad pc \triangleright \tau \cdot \lambda(r') \in \mathrm{val}_\sigma(r)\}$$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

$$\mathrm{val}_\sigma(r; s) \quad = \quad \{pc \triangleright \tau \cdot \lambda(r'; s) \mid pc \triangleright \tau \cdot \lambda(r') \in \mathrm{val}_\sigma(r)\}$$

Case Study 1: A While Language

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule (σ concrete)

$$\frac{\sigma = \mathrm{last}(\mathit{sh})}{\mathit{sh},\ \lambda(s)}$$

Software Engineering Group

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule (σ concrete)

$$\frac{\sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s)}{sh,\ \lambda(s)}$$

Case Study 1: A While Language

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule (σ concrete)

$$\frac{\sigma = \mathrm{last}(\mathit{sh}) \qquad \mathit{pc} \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad \mathit{pc}\ \text{consistent}}{\mathit{sh},\ \lambda(s)}$$

Software Engineering Group

Case Study 1: A While Language

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule (σ concrete)

$$\frac{\sigma = \mathrm{last}(\mathit{sh}) \qquad \mathit{pc} \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad \mathit{pc}\text{ consistent}}{\mathit{sh},\ \lambda(s) \to \mathit{sh} ** \tau,\ \lambda(s')}$$

Software Engineering Group

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule

$$\frac{\sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad pc \text{ consistent}}{sh,\ \lambda(s) \to sh ** \tau,\ \lambda(s')}$$

The well-formedness predicate: Nada

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule

$$\frac{\sigma = \mathrm{last}(\mathit{sh}) \qquad \mathit{pc} \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad \mathit{pc}\ \text{consistent}}{\mathit{sh},\ \lambda(s) \to \mathit{sh} ** \tau,\ \lambda(s')}$$

The well-formedness predicate: Nada

The global trace semantics for program P

$$\langle [x \to 0],\ x \in \mathrm{vars}(P) \rangle,\ \lambda(P)$$

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule

$$\frac{\sigma = \mathrm{last}(\mathit{sh}) \qquad \mathit{pc} \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad \mathit{pc} \text{ consistent}}{\mathit{sh},\ \lambda(s) \to \mathit{sh} ** \tau,\ \lambda(s')}$$

The well-formedness predicate: Nada

The global trace semantics for program P

$$\langle [x \to 0],\ x \in \mathrm{vars}(P) \rangle,\ \lambda(P) \quad \to \quad \mathit{sh}_1,\ \lambda(s_1)$$

Case Study 1: A While Language

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule

$$\frac{\sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad pc \text{ consistent}}{sh,\ \lambda(s) \to sh ** \tau,\ \lambda(s')}$$

The well-formedness predicate: Nada

The global trace semantics for program P

$$\langle [x \to 0],\ x \in \mathrm{vars}(P)\rangle,\ \lambda(P) \quad \to \quad sh_1,\ \lambda(s_1) \quad \xrightarrow{*} \quad sh_n,\ \lambda(\circ)$$

Software Engineering Group

Case Study 1: A While Language

The local evaluation rules

$$\mathrm{val}_\sigma : \mathit{Stmt} \to 2^{\mathbf{CTr}}$$

The trace composition rule

$$\frac{\sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad pc \text{ consistent}}{sh,\ \lambda(s) \to sh ** \tau,\ \lambda(s')}$$

The well-formedness predicate: Nada

The global trace semantics for program P

$$\langle [x \to 0],\ x \in \mathrm{vars}(P) \rangle,\ \lambda(P) \quad \to \quad sh_1,\ \lambda(s_1) \quad \xrightarrow{*} \quad \lim_{i \to \infty} sh_i$$

Discussion

TECHNISCHE UNIVERSITÄT DARMSTADT

Observations

- No need to represent intermediate states in rules of composite statements
- Exactly one trace generated—only one path condition can be consistent (Expected: while-language is deterministic)
- Programs allowed to diverge
- Good match with program logic based on symbolic execution

Software Engineering Group

Discussion

- No symbolic traces needed
- No events needed
- No well-formedness needed

Case Study 2: Procedure Calls

The local evaluation rules

Call a parallel procedure *m* with parameter *e*:

$$\mathrm{val}_\sigma(\mathtt{call}(m, e)) \quad = \quad \{\emptyset \triangleright \mathit{invEv}_\sigma(m, \mathrm{val}_\sigma(e)) \cdot \lambda(\text{🍶})\}$$

Case Study 2: Procedure Calls

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

Begin to execute a parallel procedure m:

$$\mathrm{val}_\sigma(m(x)\{s\}) \quad = \quad \{\ \emptyset \triangleright \mathit{invREv}_\sigma(m, Y) \curvearrowright \sigma[\qquad Y \mapsto *] \cdot \qquad \mid \\ Y \notin \mathrm{dom}(\sigma)\ \}$$

Software Engineering Group

Case Study 2: Procedure Calls

The local evaluation rules

Begin to execute a parallel procedure *m*:

$$\mathrm{val}_\sigma(m(x)\{s\}) \quad = \quad \{\ \emptyset \triangleright invREv_\sigma(m, Y) \curvearrowright \sigma[\qquad\qquad Y \mapsto *] \cdot \lambda(s[x \leftarrow y])\ |\ y,\ Y \notin \mathrm{dom}(\sigma)\ \}$$

Case Study 2: Procedure Calls

The local evaluation rules

Begin to execute a parallel procedure m:

$$\mathrm{val}_\sigma(m(x)\{s\}) \quad = \quad \{\, \emptyset \triangleright invREv_\sigma(m, Y) \curvearrowright \sigma[y \mapsto Y,\ Y \mapsto *] \cdot \lambda(s[x \leftarrow y]) \mid y,\ Y \notin \mathrm{dom}(\sigma) \,\}$$

Case Study 2: Procedure Calls

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

The trace composition rules

Continuations of parallel procedures selected from task pool (multiset) q

$$\frac{\sigma = \mathrm{last}(sh)}{sh,\ q + \{\lambda(s)\}}$$

Software Engineering Group

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

Continuations of parallel procedures selected from task pool (multiset) q

$$\frac{\sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad pc \text{ consistent}}{sh,\ q + \{\lambda(s)\}}$$

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

Continuations of parallel procedures selected from task pool (multiset) q

$$\frac{\sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(s) \qquad pc \text{ consistent}}{sh,\ q + \{\lambda(s)\} \rightarrow sh ** \tau,\ q + \{\lambda(s')\}}$$

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

Starting new task:

$$\frac{m(x)\{s\} \in \overline{M} \qquad \sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(m(x)\{s\})}{sh,\ q \rightarrow}$$

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

Starting new task: concretise method evaluation τ

$$\frac{m(x)\{s\} \in \overline{M} \qquad \rho \text{ concretises } \tau \qquad \rho(pc) \text{ consistent} \qquad \sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(m(x)\{s\})}{sh, q \to \rho(sh) ** \rho(\tau)}$$

Case Study 2: Procedure Calls

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

The trace composition rules

Starting new task: concretise method evaluation τ and extend task pool q

$$\frac{m(x)\{s\} \in \overline{M} \qquad \rho \text{ concretises } \tau \qquad \rho(pc) \text{ consistent} \qquad \sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(m(x)\{s\})}{sh,\, q \to \rho(sh) ** \rho(\tau),\, q + \{\lambda(s')\}}$$

Software Engineering Group

Case Study 2: Procedure Calls

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

The trace composition rules

Starting new task: concretise method evaluation τ and extend task pool q

$$\frac{m(x)\{s\} \in \overline{M} \qquad \rho \text{ concretises } \tau \qquad \rho(pc) \text{ consistent} \qquad \sigma = \mathrm{last}(sh) \qquad pc \triangleright \tau \cdot \lambda(s') \in \mathrm{val}_\sigma(m(x)\{s\}) \qquad wf(\rho(sh) ** \rho(\tau))}{sh,\ q \rightarrow \rho(sh) ** \rho(\tau),\ q + \{\lambda(s')\}}$$

Software Engineering Group

Case Study 2: Procedure Calls

TECHNISCHE UNIVERSITÄT DARMSTADT

The local evaluation rules

The trace composition rules

The well-formedness predicate

$$wf(sh \curvearrowright invREv(m, v)) \;=\; wf(sh) \land \#_{sh}(invEv(m, v)) > \#_{sh}(invREv(m, v))$$

Software Engineering Group

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

The well-formedness predicate

The global trace semantics for program P

$$\langle I_P \rangle,\ \lambda(P) \quad \xrightarrow{*} \quad sh,\ n \cdot \lambda(\text{🍾})$$

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

The well-formedness predicate

The global trace semantics for program P

$$\langle I_P \rangle,\ \lambda(P) \quad \xrightarrow{*} \quad \lim_{i \to \infty} sh_i$$

Case Study 2: Procedure Calls

The local evaluation rules

The trace composition rules

The well-formedness predicate

The global trace semantics for program P

Observations

- Local evaluation rules could be reused
- Minor modification to trace composition rule
- One new trace composition rule

Case Study 3: ABS

Case Study 3: ABS

Won’t fit on one slide — read the paper!

Final Remarks

TECHNISCHE UNIVERSITÄT DARMSTADT

The most modular concurrent semantics the world has ever seen?

- Presented version has statement-level process interleaving
  $\Rightarrow$ Atomic-by-default easily possible (as in ABS)
- Rules for nearly full ProMeLa, ABS in paper
- Dynamic logic calculus with soundness proof for while-language
- Executable Isabelle mechanization of most features by N. Heidler

Software Engineering Group

What is LAGC?

Lewd Artists Getting Canceled

Literally Arrogant General Computer Scientists

Louisiana Associated General Contractors (`www.lagc.org`)

Left Autoservo Gyro Control

Lazily Aggregated Gradient Coding

Lower Austrian Geo Cachers

Lovemaking Androids Grope Compulsively

Linguistically Advanced Grammatically Constrained

Laterally Askance Geometric Corpus

Ludo And his Geeky Colleagues

Liberally Applied Gherkin Cream

Last Australian Golf Club