ACP and PGA

Two Algebraic Theories in Computer Science

André van Delft

LambdaDays Krakow, 18 February 2016

Overview

- ACP - Algebra of Communicating Processes
  - Axioms
  - SubScript: GUI Application
- PGA - Program Algebra
  - Axioms
  - Java static variable initialization puzzler
- Conclusion

Algebra of Communicating Processes - 1

Bergstra & Klop, Amsterdam, 1982 - ...

ACP ~ Boolean Algebra

+	choice
·	sequence
0	deadlock
1	empty process

atomic actions a,b,…
parallelism
communication
disruption, interruption
time, space, probabilities
money
...

Algebra of Communicating Processes - 2

$$x+y = y+x$$
$$(x+y)+z = x+(y+z)$$
$$x+x = x$$
$$(x+y)\cdot z = x\cdot z+y\cdot z$$
$$(x\cdot y)\cdot z = x\cdot (y\cdot z)$$

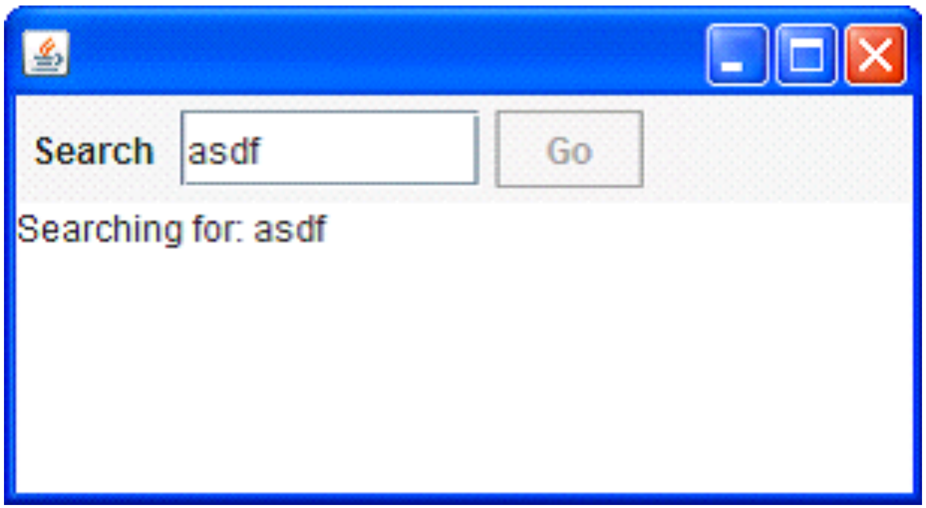
$$0+x = x$$
$$0\cdot x = 0$$
$$1\cdot x = x$$
$$x\cdot 1 = x$$

$$(x+1)\cdot y = x\cdot y + 1\cdot y$$
$$= x\cdot y + y$$

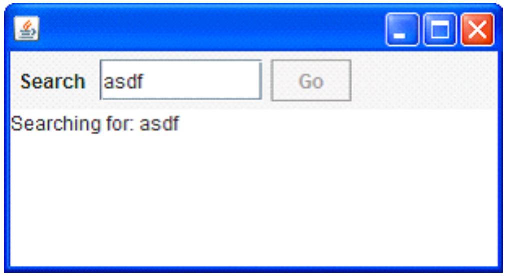
Algebra of Communicating Processes - 3

$$
\begin{aligned}
x \parallel y &= x \mathbin{⫡} y + y \mathbin{⫡} x + x \mid y \\
\\
(x+y) \mathbin{⫡} z &= \ldots \\
a \cdot x \mathbin{⫡} y &= \ldots \\
1 \mathbin{⫡} x &= \ldots \\
0 \mathbin{⫡} x &= \ldots \\
\\
(x+y) \mid z &= \ldots \\
\ldots &= \ldots
\end{aligned}
$$

SubScript: GUI application - 1

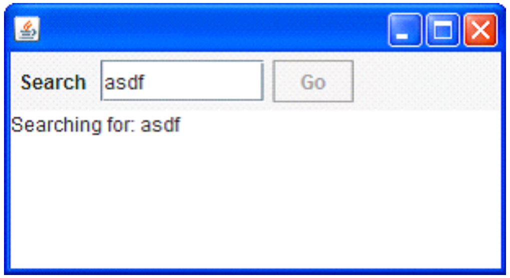


- Input Field
- Search Button
- Searching for…
- Results



SubScript: GUI application - 2

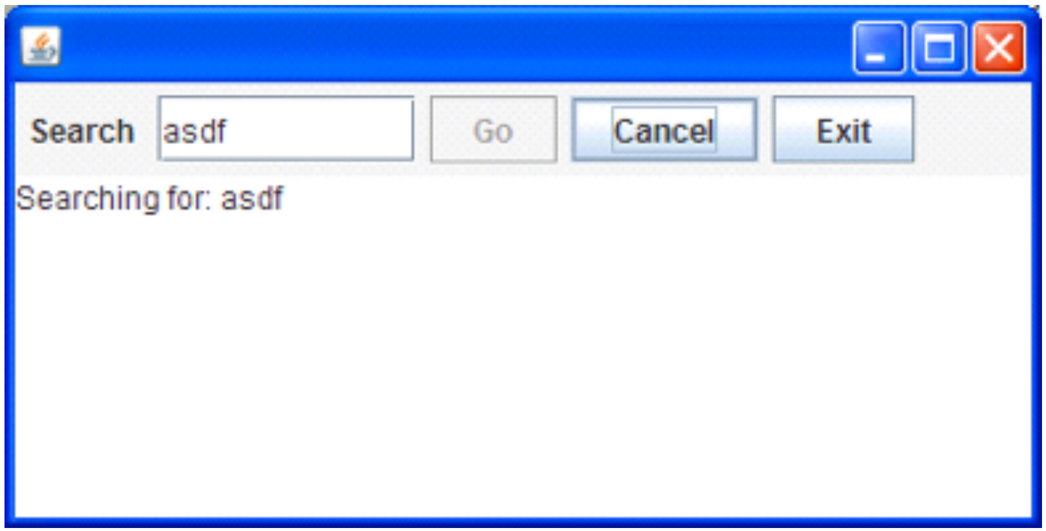
```
val searchButton = new Button("Go”) {
  reactions.+= {
    case ButtonClicked(b) =>
      enabled = false
      outputTA.text = "Starting search...”
      new Thread(new Runnable {
       def run() {
        Thread.sleep(3000)
        SwingUtilities.invokeLater(new Runnable{
          def run() {outputTA.text="Search ready”
                     enabled = true
        }})
      }}).start
  }
}
```



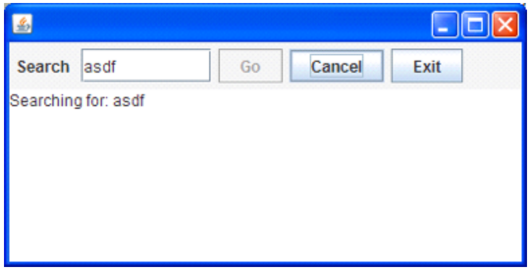
SubScript: GUI application - 3

```
live              = searchSequence...

searchSequence    = searchCommand
                    showSearchingText
                    searchInDatabase
                    showSearchResults

searchCommand     = searchButton
showSearchingText = @gui: {:outputTA.text = "…":}
showSearchResults = @gui: {:outputTA.text = "…":}
searchInDatabase  = {* Thread.sleep(3000) *}
```

SubScript: GUI application - 4



- Search: button or Enter key
- Cancel: button or Escape key
- Exit: button or  ; ; “Are you sure?”…
- Search only allowed when input field **not** empty
- Progress indication



SubScript: GUI application - 5

```
live             = searchSequence... || exit

searchCommand    = searchButton + Key.Enter
cancelCommand    = cancelButton + Key.Escape
exitCommand      =   exitButton + windowClosing
exit             =   exitCommand @gui: confirmExit ~~(b:Boolean)~~> while(!b)
cancelSearch     = cancelCommand @gui: showCanceledText

searchSequence   = searchGuard searchCommand
                   showSearchingText searchInDatabase showSearchResults
                   / cancelSearch

searchGuard      = if(!searchTF.text.isEmpty) . anyEvent(searchTF) ...

searchInDatabase = {*Thread.sleep(3000)*} || progressMonitor
progressMonitor  = {*Thread.sleep( 250)*}
                   @gui:{searchTF.text+=here.pass} ...
```

PGA - Program Algebra / Instruction Sequences 1

- Computational Model
- Alternative for Turing Machine
- Close to Assembly Language: jumps
- Sequential composition
- Axioms
- Higher levels: PGLA, PGLB, ...
- Defined using simple projections
- Applied to
  - method call dispatch in Ruby
  - static variable initialization in Java

PGA - Program Algebra / Instruction Sequences 2

Primitive instructions: for each **a** ∈ **A**, **k** ∈ **N**

- **a** - basic instruction; execution yields true/false
- **+a** - positive test instruction
  - **a** true: execute next instruction
  - **a** false: skip next instruction
- **-a** - negative test instruction
- **!** - termination instruction
- **#k** - relative jump instruction

Programs **X**, **Y**, ...

- Primitive instruction
- **X**; **Y**
- X^{ω} (= **X**; **X**; **X**; ...)

From Ponse & Van der Zwaag, *Introduction to Program and Thread Algebra*

PGA - Program Algebra / Instruction Sequences 3

Axioms for instruction sequence congruence:

$(X; Y); Z$	$=$	$X; (Y; Z)$	(PGA1)
$(X^n)^\omega$	$=$	X^ω	(PGA2, n ≥ 1, $X^1 = X$, $X^{n+1} = X; X^n$)
$X^\omega; Y$	$=$	X^ω	(PGA3)
$(X; Y)^\omega$	$=$	$X; (Y; X)^\omega$	(PGA4)

A proof of unfolding (i.e., Xω = X; Xω):

X^ω	$=$	$(X; X)^\omega$	(PGA2, n = 2)
	$=$	$X; (X; X)^\omega$	(PGA4)
	$=$	$X; X^\omega$	(PGA2)

From Ponse & Van der Zwaag, *Introduction to Program and Thread Algebra*

PGA - Program Algebra / Instruction Sequences **4**

PROJECTION ↑ ↓ EMBEDDING

PGA	
PGLA	\\#n repeat last n instructions
PGLB	backward jump
PGLC	conventional termination, without !
PGLD	absolute jump
PGLDg	numeric labels, jump-to instruction
PGLE	Instruction refinement preparation
PGLEc	If-then-else
PGLEcw	While

PGA - Program Algebra / Instruction Sequences 5

What is a program?

Answer (Bergstra, ±1998):

- A program is defined relative to a programming language
- A programming language is a pair (**L**, **ϕ**)
  with **L** a set of expressions and **ϕ** a projection to **PGA**

PGA - Program Algebra / Instruction Sequences 6

Projection semantics for multi-file programs

Jan Bergstra[*]
University of Amsterdam
Utrecht University

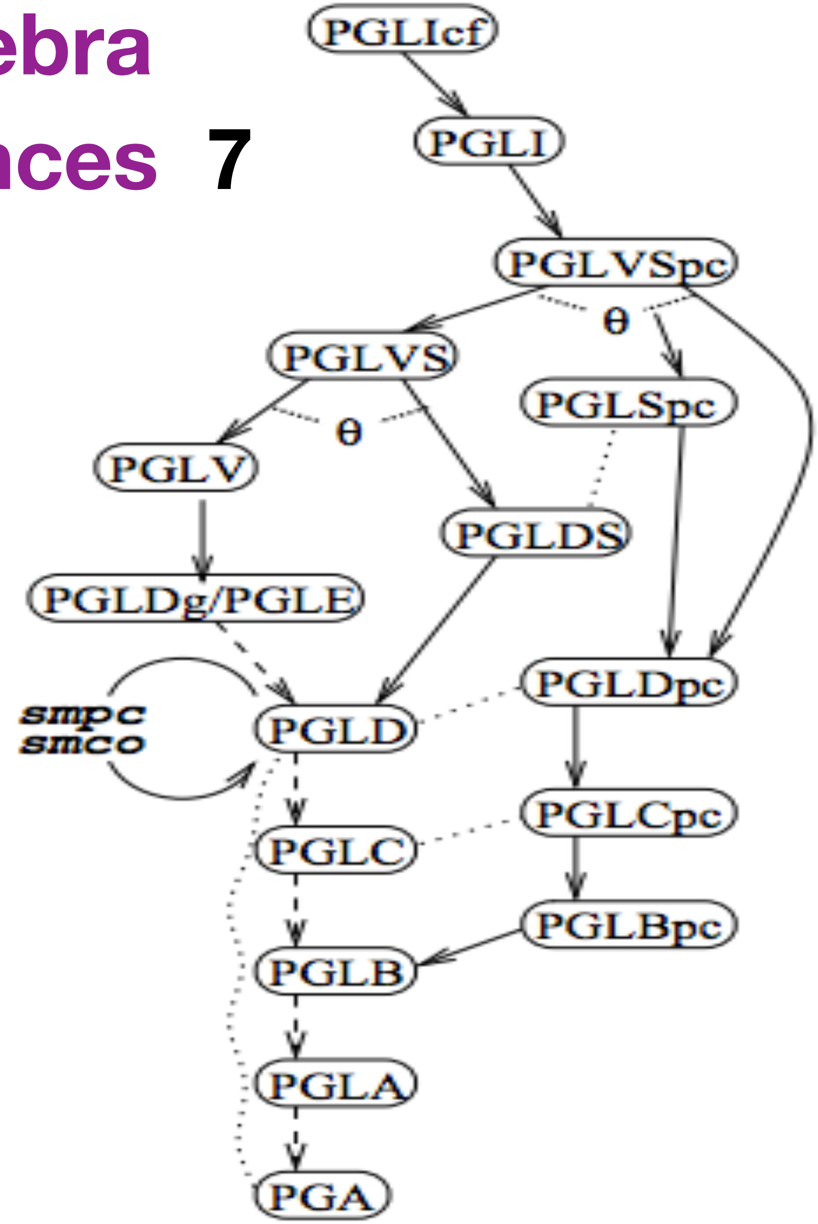
Pum Walters[†]
Microsoft

May 8, 2003

Abstract

The multi-file paradigm – where program modules are located in different files – as exhibited in Java, is investigated using the program algebra PGA. In order to so a number of auxiliary results in the context of PGA are presented: languages with explicit location of execution (PC), method invocation, structured programming, and a flat file system.

PGA - Program Algebra / Instruction Sequences 7



PGA - Program Algebra / Instruction Sequences 8

File	c0.java	c1.java	c2.java

Java Code	class c0
	{
	static void main
	(String s[])
	{
	c1.m7();
	}
	}
		class c1
		{
		static boolean b3 = c2.b5;
		static boolean b6 = true;
		
		static void m7() {
		if (b3)
		{
		b3 = false;
		c2.b5 = false;
		m7();
		}
		else
		{
		c2.m8();
		}
		}
		}
			class c2
			{
			static boolean b4 = c1.b6;
			static boolean b5 = true;
			
			static void m8() {
			System.out.println(b4);
			System.out.println(b5);
			System.out.println(c1.b6);
			}
			}

PGA - Program Algebra / Instruction Sequences 9

File	c0.java	c1.java	c2.java
PGLIcf Code	R##c2[1]; R##c1[1]; R##c1[7]; !	[1] -smbv:5; ##[2]; smbv:3.set:true; [2]; smbv:6.set:true; ##R; [7]; -smbv:3; ##[3]; smbv:3.set:false; smbv:5.set:false; R##[7]; ##[9]; [3]; R##c2[8]; [9]; ##R	[1] -smbv:6; ##[2]; smbv:4.set:true; [2]; smbv:5.set:true; ##R; [8]; *skip; ##R

PGA - Program Algebra / Instruction Sequences 10

PGLI		
R##[2,1]; R##[1,1]; R##[1,7]; !; !; !; [2,1]; -smbv:6; ##[2,2]; smbv:4.set:true; [2,2]; smbv:5.set:true; ##R;	[2,8]; *skip; ##R; !; !; [1,1]; -smbv:5; ##[1,2]; smbv:3.set:true; [1,2]; smbv:6.set:true; ##R; [1,7];	-smbv:3; ##[1,3]; smbv:3.set:false; smbv:5.set:false; R##[1,7]; ##[1,9]; [1,3]; R##[2,8]; [1,9]; ##R; !; !

Jan Bergstra

- 1951; Mathematician
- 1976-1982 Leiden University: Logic, **λ**
- 1982-2016 University of Amsterdam:
  - 1982: ACP, with Jan Willem Klop
  - 1997: Java semantics
  - 1998: PGA, with Marijke Loots e.a.
  - 2004: Promise Theory, with Mark Burgess
  - 2005: Thread Algebra
  - 2010: Proposition Algebra
- 2013: Head Informatics Section Academia Europaea

Conclusion

- **ACP**
  - Very Applicable to Programming: SubScript
  - Low Acceptance
  - Lower than CSP, CCS
- **PGA**
  - Potential Successor of Turing Machine
  - Barely Known
  - Opportunities

References

- SubScript
  - Main site: subscript-lang.org
  - Repository: github.com/scala-subscript

- Jan Bergstra
  - Personal page: staff.fnwi.uva.nl/j.a.bergstra

- PGA
  - Main site: www.science.uva.nl/research/prog/projects/pga/
  - Program algebra for sequential code: staff.fnwi.uva.nl/a.ponse/DPM/JLAP51-125-156.pdf
  - Projection semantics for multi-file programs: www.science.uva.nl/research/prog/projects/pga/pub/MF.pdf
  - A Projection of the Object Oriented Constructs of Ruby to Program Algebra: ivi.fnwi.uva.nl/tcs/pub/mastertheses/RMGeerlings.ps.gz
  - An Introduction to Program and Thread Algebra www.cs.swan.ac.uk/cie06/files/d133/c06.pdf
  - A SWOT analysis of Instruction Sequence Theory http://vixra.org/pdf/1502.0231v1.pdf