

# Flow-Based Programming



Igor Talzi

Seminar "Zehn Programmiersprachen  
in einem Semester" (CS506)

2008/15/05

# Intro

What's that?

Application development methodology and accompanying software which use a network of asynchronous, reusable, components transforming streams of formatted data objects.

- Invented back in the late '60s
- Different from conventional programming:  
subroutines → objects → data streams
- Anti-“von Neumann machine”

# Applications domains

1. E-business: banking, sale, etc.
2. Scientific apps operating with data flows.
3. Others.

# Subroutines vs. Business tasks

Subroutines:

- "square root"
- "binary search"
- "sine"
- "display a currency value in human-readable format", etc.

Operate at a single moment in time.

Business apps building blocks:

- "merge"
- "sort"
- "paginate a report", etc.

Operate over a period of time, across a (large) number of input and output items.

# Major features

- “*Configurable Modularity*”: **component-oriented**. Applications are defined as networks of "black box" processes, which exchange data across predefined connections.
- Reusability, patterning.
- Parameterization.
- Flows of data are being processed **asynchronously**. Timing is unpredictable. Event-driven architecture.
- Focus on transformations performed on streams of data instead of sequences of actions.
- A good fit with **multiprocessor** systems.
- Takes the best principles of distributed systems and OOP: “*active objects*”.

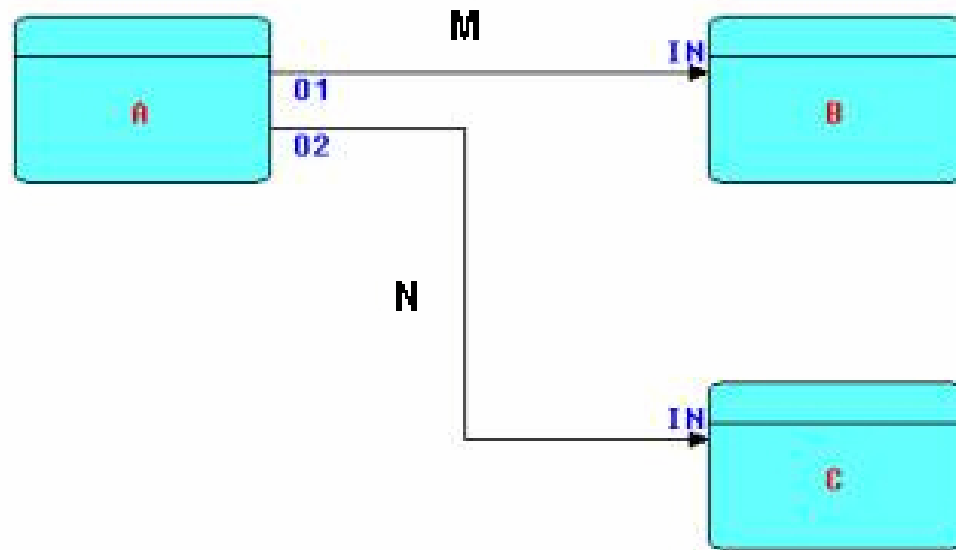
# Contestants / Ancestors

- Higher-Level Langs (HLLs):
  - ✓ COBOL (business)
  - ✓ Algol (algorithms)
  - ✓ C (all-purpose)
  - ✓ etc.
  - 1- Poor logic support (IF, FOR, WHILE, SWITCH)
  - 2- Existing logic is here for sync of data
- 4G Langs: pattern-based HLL generators
  - 1- Opaqueness
- CASE tools

# Major Terms

- Components (elementary/composite)
- Packages
- Processes
- Fixed-capacity connections (**FIFO** queues)
- I/O ports (connections  $\leftrightarrow$  processes)
- IPs – Information Packets (**must be disposed of, NO garbage collection**)
- Data streams (keys vs. brackets), substreams
- Scheduling

# Simple FBP diagram and an IP stream



$x_1 [y_1 z_{11} z_{12} z_{13}] [y_2 z_{21} z_{22} z_{23}] x_2 [y_3 [z_{31} z_{32}]]$

# Quick dictionary

Operations and events:

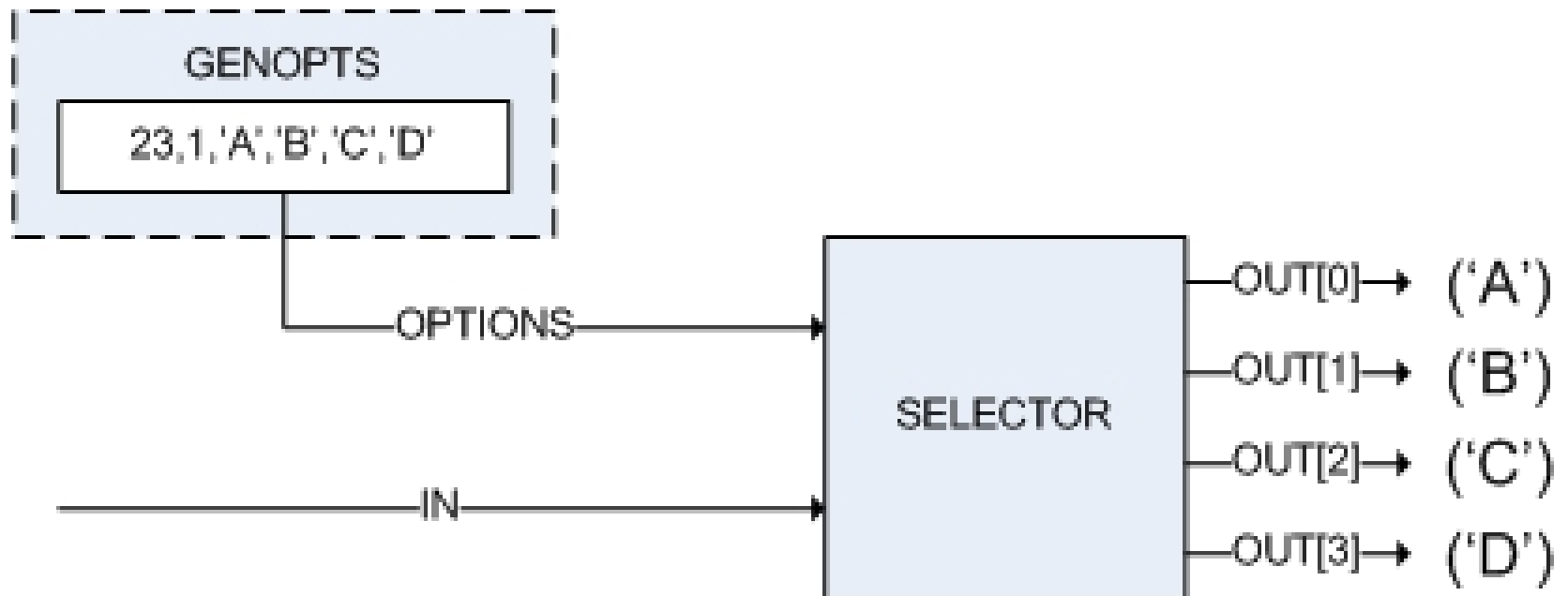
- "read" → "receive"
- "write" → "send"
- "discard" → "drop"
- "end of file" → "end of data"
- Typical components groups:  
*sort, collate, split, assign, replicate, count, concatenate, compare, generate reports, read, write, transform, manipulate text, discard, dumper, look-up tables.*

# Reusability

- Components are specified as “black-boxes” with precisely defined interfaces.
- Generalized (sorters, comparators) and “technology-dependent” (read, write) components.

# Parameterization

- It's just an IP (**IIP** – see next slide).
- Parameters can be generated internally at run-time.

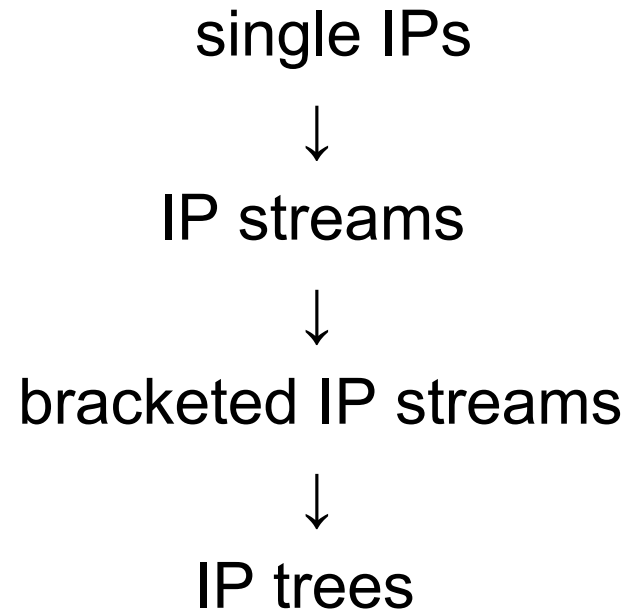


# IPs

- Data
- Control (substream control stack)
- Data/Control
- Service (e.g. count)
- Initial (for parameterization, connected to the “OPTIONS” port)

Can be in a free-form or can have a fixed layout.

# IP trees

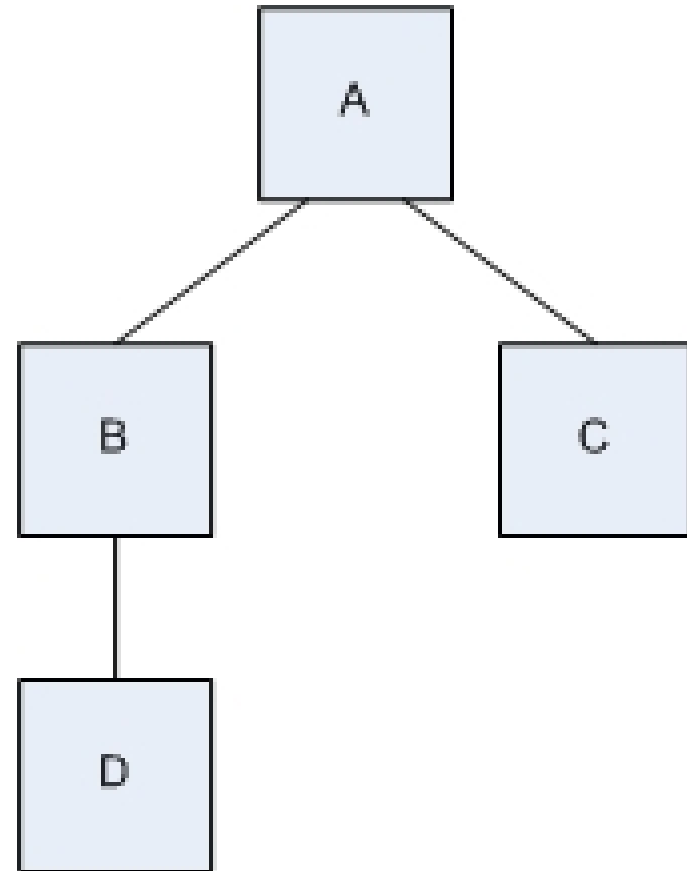


**Only handles travel through the net!**

# IP trees (cont.)

- IP tree – a structure of linked IPs.
- Can be assembled/ disassembled on intermediate nodes.
- No multi parental relationships, no loops.
- Can be replaced with substreams (LISP “list of lists”):

**[A [B D] C]**



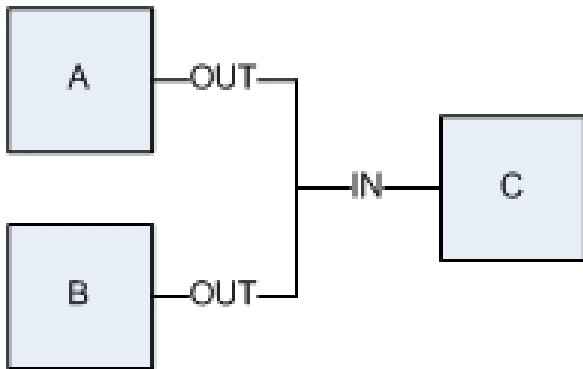
# Dynamic subnets

- Are similar to dynamically loadable modules.
- (Un-)loaded, driven and monitored by a Subnet Manager (“mother”-process) enveloping the entire subnet.
- Subnets hierarchy (levels) can be matched to bracketed substreams.

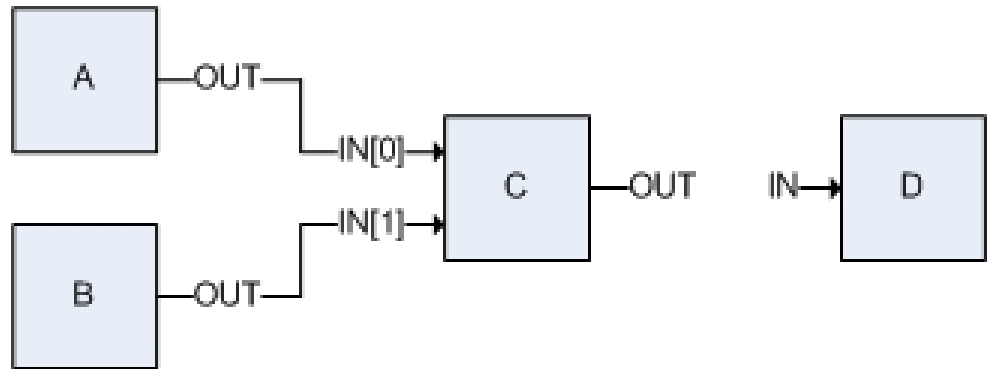
# Split & Merge

- No problem with splitting, or "divergent" patterns, or desync.
- How about merging?

simple merge



collate/concatenate



# Descriptors

- Metadata specifying IPs' layout.
- Descriptors are attached to IPs.
- Allows different representations of the same data:

**019920131F = \$199,201.31 = 31st January, 1992**

- Can be extended with dynamic attributes:  
*null, modified, etc.*

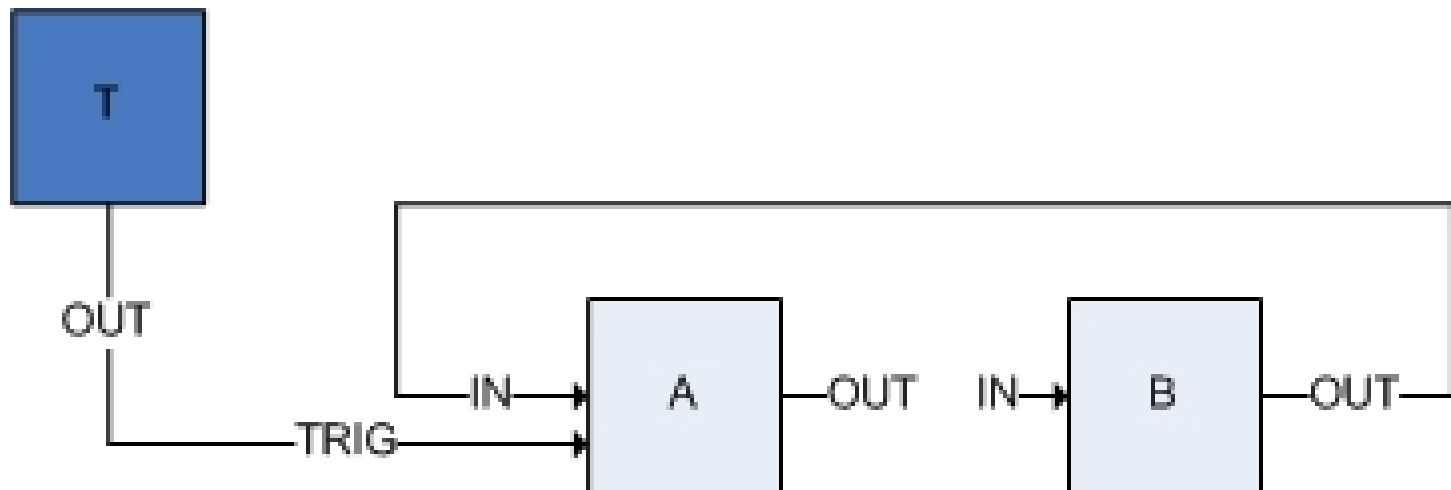
# Scheduling

- IPs are processed upon arrival (if the component is free).
- Components: loopers and non-loopers.
- Only process with no input connections start automatically.
- Process states: **initiation** → active/inactive ...  
active/inactive → **termination**  
active: normal and suspended

# Loop-type networks

Are useful for parsing recursive structures (e.g. list of lists, reg expressions, etc).

- How to get started? “Kicker”
- How to close down? “catch-22” situation  
close one of I/O ports
- Sync/capacity issues use IP trees or substreams



# Deadlocks

1. trans A - get X with hold
2. trans B - get Y with hold
3. trans B - try to get X with hold
4. trans A - try to get Y with hold
5. ...

Between processes, and between networks!

Deadlock types:

1. "resource deadlock" (or "dining philosophers problem")
2. "infrastructure" or "hybrid" deadlocks (blocking on the external signal)
3. "storage deadlock"
4. "resource deadlock"-2 (send more than receive)

# Synchronization

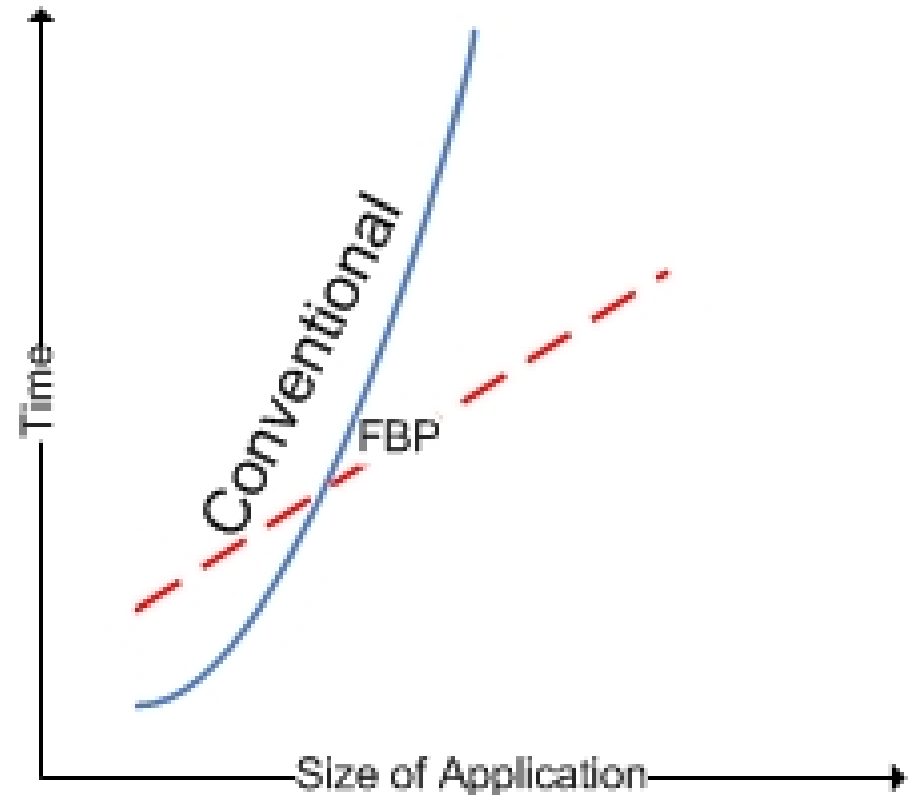
- Waiting on events (generated and consumed by processes).
- Sync to an absolute time.
- Delays.
- Monitoring (dynamic subnets).

# FBP vs. OOP

- FBP has a lot in common with OOP.
- Both use “handles”.
- FBP features: genericity (single IP, multiple results), inheritance (descriptors and attributes), encapsulation (IP filtering, network level protection).
- Message passing = IP streams.
- Most OO implementations are sync (e.g. RPC).

# Advantages

- Rapid prototyping
- Reliable and maintainable systems
- Consistency
- Scalability
- Visualization



# Performance

- Real-time requirements are hardly reachable.
- Performance highly depends on the design ("granularity").

# Tools

Implementations of the FBP concepts:

- JavaFBP
- THREADS (C)
- C#FBP
- DrawFBP

# JavaFBP

Go to the JavaFBP webpage...

# References

- Official page:  
<http://www.jpaulmorrison.com/fbp/index.shtml>
- Sourceforge link for the tools:  
<http://sourceforge.net/projects/flow-based-pgmg>
- “Flow-Based Programming”, van Nostrand Reinhold, 1994.
- Wikipedia

# Exercises

- See the PDF description...

Thanks for your attention!

Questions?