Towards a Dataflow Approach to Robot Programming

Orestis Melkonian

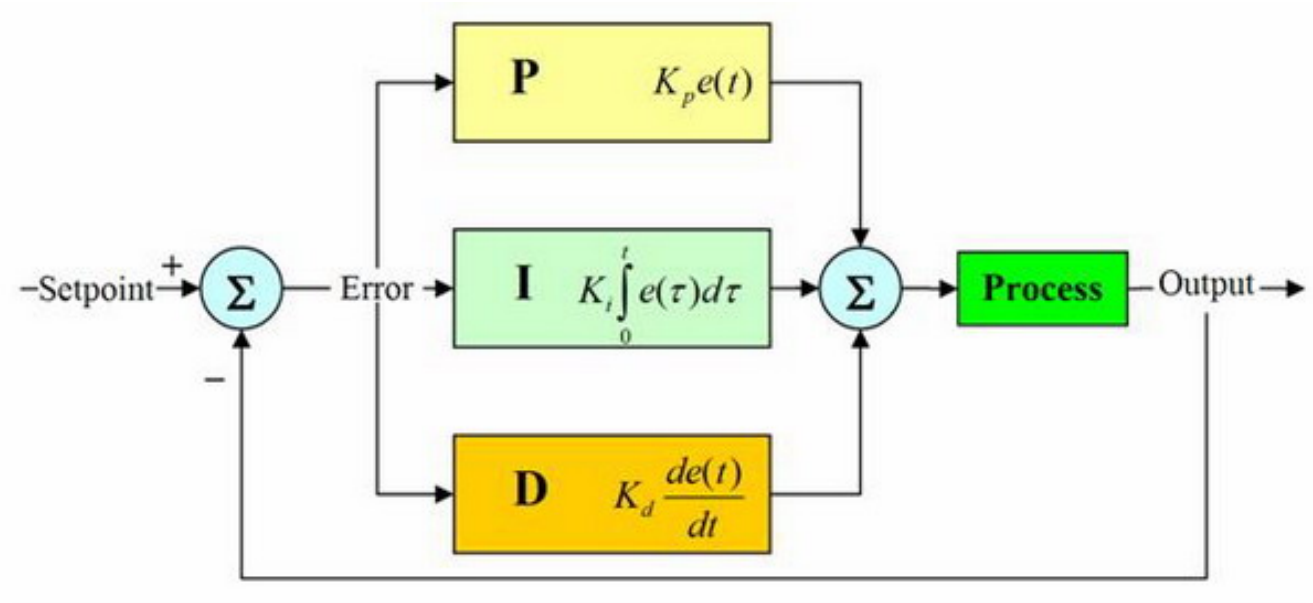
Software & Knowledge Engineering Laboratory (SKEL)

NCSR "Demokritos"

Overview

Motivation I

- **Common Patterns**
  - Robot perception architecture
  - Feedback loop controllers



Motivation II

- **ROS**: Robot Operating System
  - Hardware abstraction
  - Reusability
  - Language-agnostic open-source middleware
  - Publish-Subscribe design pattern
  - Communication via "topics"
  - Status Quo
    - Almost all code written in C++ and Python
    - Callbacks
    - Dataflow nature so far ignored

Stream Framework

- Topics as streams
- At a micro-level, replace callback "internal plumbing" with clean functional declarations
- At a macro-level, acts as a coordinating language adding to the composability of ROS
- **Extensibility**
  - Strategy design pattern for evaluation
  - Coder simply declares a dataflow graph
- **Advantages**
  - Decouple design (what to do) from execution (how to do it)
  - Cleaner, easier to maintain code
  - Implicit concurrency
  - Implicit message-passing

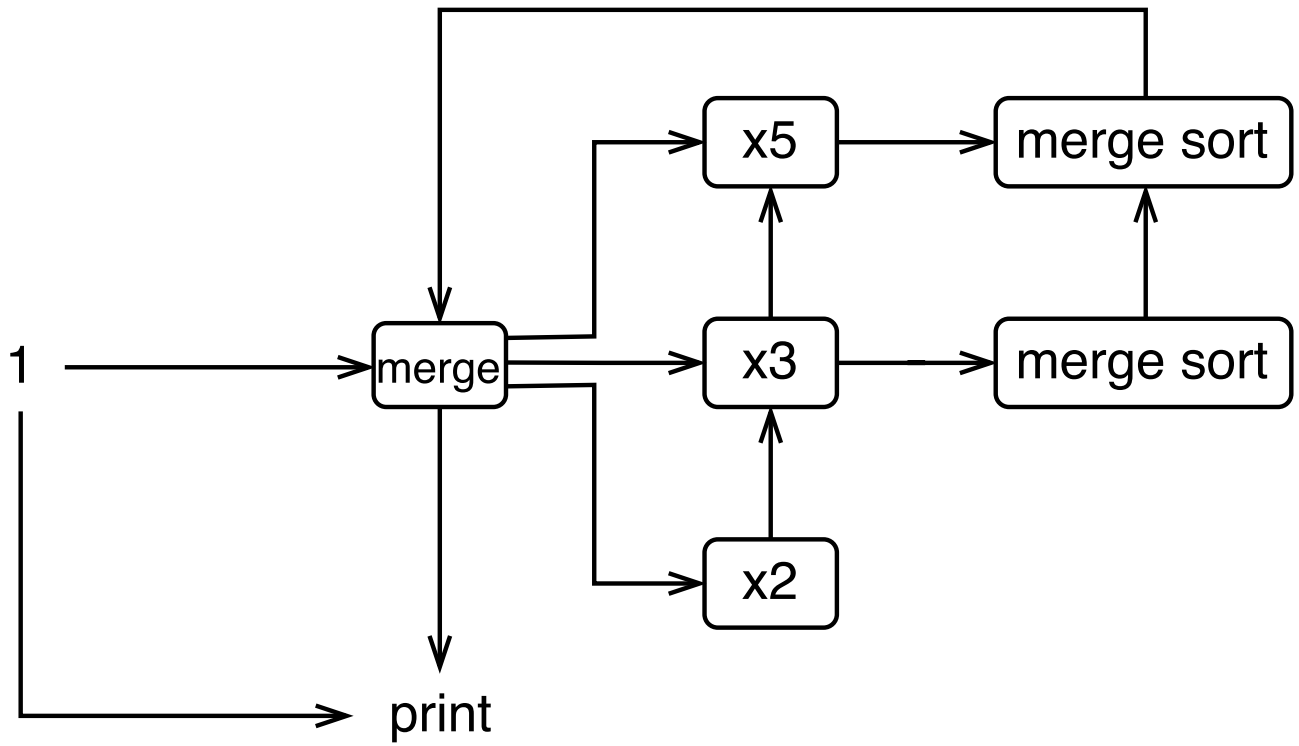
Future Work: Optimization

- General graph transformations
  - Apply some simple heuristics
  - Preserve semantics
  - Back-ends continue with more specific optimizations
- Network-aware placement
  - Fusion/fission to reach desired granularity
  - Dynamic reconfiguration

Future Work: DSL

- Minimize boilerplate code
- More intuitive syntax
- Embedded in Scala
  - Functional
  - Inherit rich type system
  - Little programming effort
- Restrict host language
  - Single-assignment
  - Restricted resource usage
- Impose a specific program structure
  - Minimize design flaws

Demos: Hamming Numbers ($2^i 3^j 5^k$)



The End