A Domain-Specific Language for Dynamic Resource Management

Resource Management

- Required for device initialization
  - Initializing a device requires resources
  - An initialized device provides resources
- Simple case handled well
  - Single level of service
  - Largely persistent
- Complex cases ignored
  - Dependable service levels
  - Removable devices

The Problem

- Complex resource dependencies
  - Variable quality of service
  - Multiple levels of resource dependency
  - Suggests constraint solving
- Hierarchical resource relationships
  - Lots of repetition in resource descriptions
  - Suggests a type system
- Dynamic resource addition and removal
  - Not handled well by current systems
  - Need to solve reverse dependencies
  - Suggests a non-imperative language

Related Work

- Barrel Fish
  - Constraint solver for device initialization
  - Missing sub-typing
- Device Tree
  - Unified device naming scheme
  - Missing constraint solving
- Tessellation
  - Resources provided by software
  - Supports QOS guarantees

DSL

- Type system
  - Solves the repetition problem
  - Some syntactical sugar added (lexographical subtyping, additional arithmetic expressions)
  - Integrated with constraint solving
- Pattern matching
  - Simple expression of complex resource relationships
  - Most of the runtime deals with pattern matching

Implementation

- DSL Runtime
  - ~1000 lines of C code
  - Uses a linear integer constraint solver (minion)
- Test case for Linux
  - Loads the PCI device tree from /sys
  - Solves an allocation in ~30ms

Further Questions

- Cost minimization
  - Supported by the solved
  - How to integrate cleanly into language?
- Scheduling
  - Requires adding execution
- Solution stability
  - Moving devices can be expensive
  - Maybe solvable via cost minimization
- Optimization
  - Apply compiler optimization techniques to the constraint generator

References