A Graph Transformation Approach to Architectural Run-Time Reconfiguration

Michel Wermelinger   Antónia Lopes   José Luiz Fiadeiro

Laboratório de Modelos e Arquitecturas Computacionais
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1700 Lisboa, Portugal

Introduction

Motivation

- systems evolve: new requirements or new environment (failures, transient interactions)
- for safety or economical reasons, some systems cannot be shut off to be changed
- domain with some interest in SA community but little formal work

Issues

time before or at run-time (dynamic reconfiguration)

source user (ad-hoc); topology or state (programmed)

operations add/delete components/connections; query topology/state

constraints structural integrity; state consistency; application invariants

specification architecture description, modification, constraint languages

management explicit/centralised (configuration manager); implicit/distributed (self-organisation)

Related Work

- Distributed Systems, Mobile Computing, Software Architecture
- not at architectural level
- not arbitrary reconfigurations
- low-level behaviour specification (process calculi, term rewriting, etc.)
- interaction between computation and reconfiguration: complex, implicit, or blurred
- tool support, in particular automated analysis

Approach

- use parallel program design language with state for computations
- category of programs with superposition
- architecture = categorical diagram; system = colimit
- architecture = graph; reconfiguration = rewriting
- apply algebraic graph transformation
  - uses category theory
  - much work done on it
  - double-pushout approach avoids side-effects
- conditional rules to add/remove components/connectors
- typed graphs for reconfiguration-invariant architectural type

Advantages

- expressive, simple, uniform, explicit, algebraic framework to specify dynamic reconfiguration
- diagrams represent connectors, architectures, reconfiguration rules, and architectural types in graphical yet mathematical rigorous way
- colimits to obtain connector semantics, systems, reconfiguration steps and to relate explicitly computation and reconfiguration
- simple higher level program design language with intuitive state representation
- handle state transfer and removal/addition in correct state
- simple, declarative constraints on possible interactions

Position

- **run-time** reconfiguration is an important issue for current software systems
- need formal approach at **high level** of abstraction to support design and analysis
- categorical framework allows to relate **heterogeneous** formalisms