Interface Theories for Component-based Design

Luca de Alfaro Thomas A. Henzinger

compositionality seminar presentation by Paul Rehrl

Introduction

- interface models $\longrightarrow$ compositional abstraction
- component models $\longrightarrow$ compositional refinement
- block diagrams to model system structure
- interface theories
    - stateless interfaces
    - stateful interfaces

Interfaces vs. Components

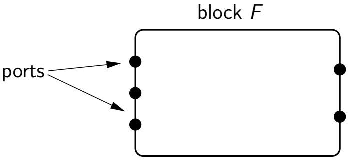
- interface models
  - interfaces constrain the environment
  - interface description answers: *How can it be used?*

Interfaces vs. Components

- interface models
  - interfaces constrain the environment
  - interface description answers: *How can it be used?*
- component models
  - components do not constrain the environment
  - component description answers: *What does it do?*

Block Diagrams

Block diagrams are a general way of depicting system structure.

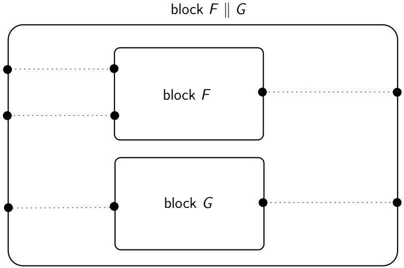


set P_F of ports

port x has type $\mathbb{T}_x$

Composition

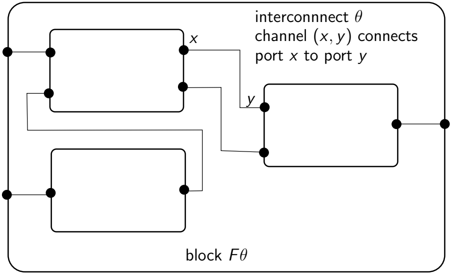
Composition is a partial binary function $\|$ on blocks.



$P_{F\|G} = P_F \cup P_G$

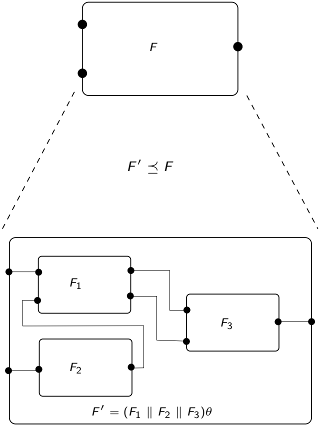
Connections

A connection is a partial function mapping a block F and an interconnect θ to a block $F\theta$.



Hierarchy

- hierachy relation $\preceq$ on blocks
- $\preceq$ is reflexive and transitive
- $F' \preceq F$
  - block F' refines block F
  - block F abstracts block F'



Interface Algebra

A block diagram is a interface algebra if the blocks F, G, and F' are interfaces and F' refines F then

- $F' \parallel G$ refines $F \parallel G$
- $F'\theta$ refines $F\theta$

Component Algebra

A block algebra is a component algebra if the blocks f, g, and f' are components and f abstracts f' then

- $f \parallel g$ abstracts $f' \parallel g$
- $f\theta$ abstracts $f'\theta$

Implementation

Give a interface algebra $\mathcal{A}$ and a component algebra $\mathcal{B}$, an implementation of $\mathcal{A}$ by $\mathcal{B}$ is a relation $\triangleleft$ between components of $\mathcal{B}$ and interfaces of $\mathcal{A}$.

Implementation

Give a interface algebra $\mathcal{A}$ and a component algebra $\mathcal{B}$, an implementation of $\mathcal{A}$ by $\mathcal{B}$ is a relation $\triangleleft$ between components of $\mathcal{B}$ and interfaces of $\mathcal{A}$.

The implementation $\triangleleft$ is compositional if

- interfaces F and G are composeable and the components f, g implement F, G then $f \parallel g$ is defined and $f \parallel g$ implements $F \parallel G$
- if f implements F then $f\theta$ implements $F\theta$ for all interconnects θ

Interface Theory

If interface algebra $\mathcal{A}$ implements component algebra $\mathcal{B}$ and $\lhd$ is a compositional implementation of $\mathcal{A}$ by $\mathcal{B}$ then $(\mathcal{A}, \lhd)$ is a interface theory for $\mathcal{B}$.

Compositional Design

Interface theories support compositional design:

- F is split into $(F_1, \ldots, F_n)\theta$
- interfaces F_i can be implemented independently
- $(f_1, \ldots, f_n)\theta$ implements F

Component Verification

Component algebras support component verification:

- f is some property of the component
- $(f_1, ..., f_n)\theta$ satisfies f
- every f_i can be verified independently

Some Interface Algebras

- input/output interfaces
  - input and output ports
  - ports are typed

Some Interface Algebras

- input/output interfaces
  - input and output ports
  - ports are typed
- assume/guarantee interfaces
  - I/O interface
  - constrains input value and output value ranges

Some Interface Algebras

- input/output interfaces
  - input and output ports
  - ports are typed
- assume/guarantee interfaces
  - I/O interface
  - constrains input value and output value ranges
- port-dependency interfaces
  - I/O interface
  - constrains which output ports may influence which input ports
  - provides dependency information between input ports and output ports

Input/Output Interface Algebras

composition: defined if output ports do not overlap

connection: only connections from output ports to input ports
no two channels have the same target

hierarchy: if F' implements F then
F' may use all input ports of F
must provide values for all output ports of F

Assume/Guarantee Interface Algebras

composition: corresponding I/O interface composable
input assumptions and output guarantees of F and G are merged

connection: I/O interface is connectable
input assumption of $F\theta$ is satisfiable
output guarantees of F are not violated

hierarchy: if $F' \preceq F$ then F' must accept all inputs that satisfy the input assumption of F and may only produce outputs that satisfy the output guarantee of F

Port-dependency Interface Algebras

composition: I/O interface is composable

connection: I/O interface is connectable
port dependencies introduced by the interconnect must not lead to a dependency cycle

hierarchy: a refined interface F' must not have more dependencies than permitted by F

Component Algebras

For every interface algebra we give an example of a component algebra such that there is a compositional implementation.

All component algebras presented in the paper are relational nets, i.e. set of blocks called processes which are connected by channels.

Relational Nets

A relational net is well-formed if

- each process specifies a relation between input and output ports
- there exists port values that satisfy the I/O relations of all processes and identities enforced by channels

Component Algebras

no interface theory for relational nets, but interface theories for restricted classes:

Component Algebras

no interface theory for relational nets, but interface theories for restricted classes:

- rectangular nets
  - processes restrict the accepted input values
  - no I/O dependencies

$\longrightarrow$ assume/guarantee interfaces

Component Algebras

no interface theory for relational nets, but interface theories for restricted classes:

- rectangular nets
  - processes restrict the accepted input values
  - no I/O dependencies

  $\longrightarrow$ assume/guarantee interfaces
- total nets
  - processes do not restrict the accepted input values
  - I/O dependencies

  $\longrightarrow$ port-dependency interfaces

Component Algebras

no interface theory for relational nets, but interface theories for restricted classes:

- rectangular nets
  - processes restrict the accepted input values
  - no I/O dependencies

  $\longrightarrow$ assume/guarantee interfaces
- total nets
  - processes do not restrict the accepted input values
  - I/O dependencies

  $\longrightarrow$ port-dependency interfaces
- total-and-rectangular nets
  - no restrictions on input values
  - no I/O dependencies

  $\longrightarrow$ input/output interfaces

Stateful Interfaces

A stateful interface proceeds in steps through a state space.

- deterministic assume/guarantee interfaces
  - input assumption and output guarantee depend on state
  - state transition function

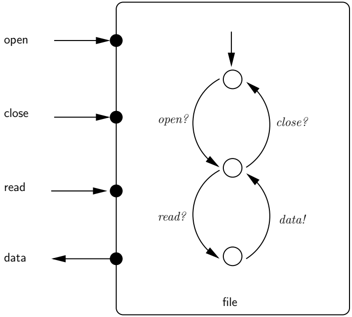
Stateful Interfaces

A stateful interface proceeds in steps through a state space.

- deterministic assume/guarantee interfaces
  - input assumption and output guarantee depend on state
  - state transition function
- deterministic game interfaces
  - a set of states and a set of ports
  - input moves and output moves for every state

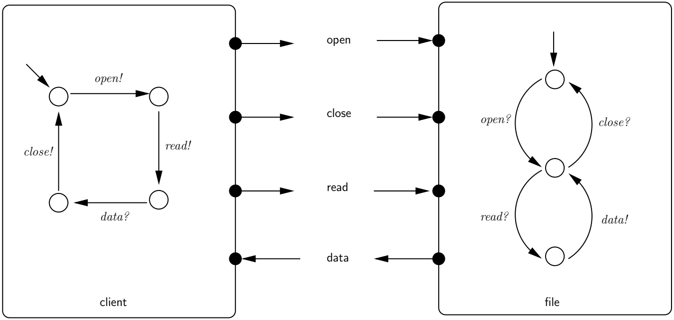
Stateful Interfaces

Example: file



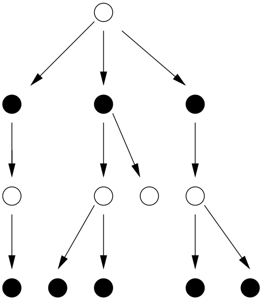
Stateful Interfaces

Example: file



Stateful Interfaces

- stateful interfaces are viewed as a game between the environment and the component
- the environment looses if the game enters a state in which the environment can not provide acceptable input (error state)



end of presentation