Slicing of Component Behavior Specification w.r.t. their Composition

Ondřej Šerý and František Plášil

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://dsrg.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
Goal

• Motivation
  - Observation: “Reusable SW components usually provide more functionality than actually used in a concrete architecture/assembly”
    - Behavior specification of such components is overspecified
    - Need for automatic slicing of the unused behavior, to
      - reveal actual roles of the components
      - make understanding of spec easier

• Goal
  - Slicing of behavior spec with respect to composition
Motivation example 1: An airport internet providing application
Motivation example 2: Reuse in a public garden…
Database frame protocol

```java
?db.start{!logger.start ; !tm.init} ;
{
  ?db.add{!tm.begin ; (!tm.commit + !tm.rollback)}
  ||
  ?db.get{!tm.begin ; (!tm.commit + !tm.rollback)}
  ||
  ?db.remove{!tm.begin ; (!tm.commit + !tm.rollback)}
}*
?db.stop{!logger.stop ; !tm.destroy}
```
Behavior Protocols – syntax

- Behavior protocol
  - Expression describing the behavior of a software component
    - Infinite set of finite event traces

- Events:
  - Emitting a method call request: \(!interface\text{.}method\uparrow\)
  - Accepting a method call request: \(?interface\text{.}method\uparrow\)
  - Emitting a method call response: \(!interface\text{.}method\downarrow\)
  - Accepting a method call response: \(?interface\text{.}method\downarrow\)

- Operators:
  - Sequence: \(\;\)
  - Alternative: \(+\)
  - Repetition: \(*\)
  - And-parallel interleaving: \(\mid\)
  - Or-parallel interleaving: \(||\)
  - Consent \(\nabla\)
    - parallel composition (interleaving + \(\tau\))
    - indicating communication errors
    - no activity (deadlock)
    - bad activity (! cannot be responded)

- Syntactic abbreviations (to express method calls)
  - \(?i.m = ?i.m\uparrow ; !i.m\downarrow\)
  - \(?i.m\{prot\} = ?i.m\uparrow ; prot ; !i.m\downarrow\)
Behavior Compliance

- **Horizontal compliance**
  - $\text{Database}_F \mathbin{\triangledown} \text{Logger}_F \mathbin{\triangledown} \text{Transactionmanager}_F = \text{Architecture}_\text{prot}$
  - Tool: Behavior protocol checker (BPC) - Checks for communication errors

- **Vertical compliance**
  - $\text{Architecture}_\text{prot} \mathbin{\triangledown} \text{DBServer}_F^{-1}$
  - Tool: Behavior protocol checker (BPC) - Checks for communication errors
Substitutability – Example

• B is substitutable for A
  - ?b is not used in the given environment

• C is not substitutable for A
  - Bad-activity
  - Environment may not expect !y call

• D is not substitutable for A
  - No-activity (deadlock)
  - Environment may wait for !x
**Substitutability**

- **Def:** Protocol $a$ is *substitutable* for $b$ if $L(a \lor b^{-1})$ does not contain any communication error.

- $B^{-1}$ is the most general environment of $B$.

---

**Diagram:**

- **Case a:**
  - $B$ with frame prot. $b$.
  - $B^{-1}$ with frame prot. $b^{-1}$.

- **Case b:**
  - $B$ with frame prot. $b$.
  - $Env_B$.

- **Case c:**
  - $A$ with frame prot. $a$.
  - $B^{-1}$ with frame prot. $b^{-1}$.

---

Ondřej Šerý and František Plášil: *Slicing of Component Behavior Specification with Respect to Their Composition*, CBSE 2007
Reduction

• Motivation ($A_{\text{orig}}$ is overspecified)
  - Replacing $A_{\text{orig}}$ by $A_{\text{red}}$
    - $A_{\text{red}}$ is a reduced variant of $A_{\text{orig}}$
  - Guarantee that:
    - Reasoning about $A_{\text{red}} \sqcap \text{Environment}$ will also apply to $A_{\text{orig}}$

$\rightarrow A_{\text{orig}}$ has to be substitutable for $A_{\text{red}}$

$A_{\text{red}} = \text{reduced variant of } A_{\text{orig}}$
Reduction

• To put it more formally…

• **Def:** A protocol $X$ is a *reduction* of $Y$ if
  - $Y$ is substitutable for $X$
  - $L(X) \subseteq L(Y)$
  - Intuitive extension to a *minimal reduction*
Slicing

• Why slicing (is reduction not enough)?
  ▪ Reduction considers only semantics not syntax
    • Fewer behavior ≠ simpler protocols

→ Slicing – based on pruning the syntax trees
  ▪ Resulting protocols are syntactically simplified

• Def: A protocol $X$ is a slice of $Y$ if
  ▪ $Y$ it is substitutable for $X$
  ▪ The syntax tree of $X$ can be derived by pruning the syntax tree of $Y$
Slicing w.r.t. composition

Components and protocols

Syntax trees

Automata

Communication (∨)

Ondřej Šerý and František Plášil: Slicing of Component Behavior Specification with Respect to Their Composition, CBSE 2007
Implementation

• Implemented as an extension to the behavior protocol checker – dChecker:
  ▪ Available at:
    http://dsrg.mff.cuni.cz/projects/dchecker

• Tested on the demo application
  ▪ Developed as a part of the Fractal Component Reliability Extension project:
    http://fractal.objectweb.org/fractalbpc
Case study

- Airport scenario
  - Frequent flyers
  - Business + 1st class passengers
  - Payment by a credit card

- Public garden scenario (reuse)
  - Only a credit card payment

Protocol characterizing the environment:

```plaintext
(?
  ILogin.GetTokenIdFromIpAddress +
  ILogin.LoginWithFlyTicketId +
  ILogin.LoginWithFrequentFlyerId +
  ILogin.LoginWithAccountId +
  ILogin.Logout +
  IAccount.GenerateRandomAccountId +
  IAccount.CreateAccount +
  IAccount.RechargeAccount
) *
```
Reducing protocol (1)

Arbitrator

```plaintext
(:
  ?ILogin.GetTokenIdFromIpAddress +
  ?ILogin.LoginWithFlyTicketId {
    !IFlyTicketAuth.CreateToken_1 ;
    (!IFirewall.DisablePortBlock + NULL)} +
  ?ILogin.LoginWithFrequentFlyerId {
    !IFrequentFlyerAuth.CreateToken ;
    (!IFirewall.DisablePortBlock + NULL)} +
  ?ILogin.LoginWithAccountId {
    !IAccountAuth.CreateToken ;
    (!IFirewall.DisablePortBlock + NULL)} +
  ?ILogin.Logout {!IToken.InvalidateAndSave_1 + NULL}
)* |
?ITokenCallback.TokenInvalidated_1 {!IFirewall.EnablePortBlock_1}* |
?ITokenCallback.TokenInvalidated_2 {!IFirewall.EnablePortBlock_2}* |
?ITokenCallback.TokenInvalidated_3 {!IFirewall.EnablePortBlock_3}* |
?IDhcpCallback.IpAddressInvalidated {
  !IToken.InvalidateAndSave_2 + NULL}*
)
Reducing protocol (2)

DHCPServer:

```plaintext
!IDhcpCallback.IpAddressInvalidated*
|
|
{?
?IManagement.UsePermanentIpDatabase^ ; (  
   !IIpMacPermanentDb.GetIpAddress*
   |  
   (  
      !IManagement.UsePermanentIpDatabase$ ;  
      ?IManagement.StopUsingPermanentIpDatabase^  
   )  
) ; !IManagement.StopUsingPermanentIpDatabase$
}*
```
Related work

• Program slicing (traditional)
  ▪ Finding a “slice” – a minimal form of a program exhibiting the behavior of interest
  ▪ Based on the control and data flow analysis
  ▪ Debugging, Software maintenance, Model Checking (state space reduction), …

• General slicing
  ▪ Extending the idea to a general algorithm working on the syntax-tree of an expression
Related work

• Requirement specification slicing
  ▪ Applying slicing on the specification-level

• Architectural slicing
  ▪ Applying slicing on the architectural-level

• And other…
Conclusion

• We propose slicing of behavior specification
  → Simple straightforward technique:
    *slicing with respect to composition*

• Prototype implementation
  ▪ Extension of the behavior protocol checker

• Case study
  ▪ Demonstrating the technique on the demo airport/garden application
Questions...?

Thank you for your attention

Any questions?

• Answers also at: http://dsrg.mff.cuni.cz