Incremental Analysis of Interference Among Aspects

Authors:
Emilia Katz, Shmuel Katz

The Technion

Motivation

- Multiple aspects are often woven into the same system

  => Unintended interactions among the aspects may occur, even if each aspect is “correct” when woven alone
- Libraries of reusable aspects (example: a library implementing the ACID properties for transactional objects)

  => Usage guidelines for the participating aspects are needed

New Interference Type

Previously defined interference types:

Interference caused by -

- Common join-points
- Updating shared variables
- Changing join-points

example – soon!

⇒Not enough!

⇒More general definition is needed!

Interference caused by the **semantics** of the aspects!

What is a "correct" aspect?

Aspect Specifications

prior

… because model-checking is used in proof method automatization …

Pair of **LTL** formulas

Specification of aspect A is **(P_A, R_A)**

The principle: **assume – guarantee** (generalized)

A assumes: P_A holds in the base system

in **any reasonable** base system for A

- what's true at joinpoints
- global properties of base system
- properties of aspect parameters

unusual!

in **any** woven system with A

A guarantees: R_A is true in the woven system

- new properties added by A
- properties of base system maintained in woven system

possibly global !

Semantic Interference Among Aspects

pairwise definition; will be generalized to N aspects…

One aspect “causes” another to not give the desired result (violate its guarantee):

- Aspect A satisfies its specification **(P_A, R_A)**
- Aspect B satisfies its specification **(P_B, R_B)**
- Base system satisfies both **P_A** and **P_B**

From now on: assume all the aspects are "correct"

Aspect Interference

A, B – aspects; S – underlying system

S + A ➔ OK

S + B ➔ OK

(S + A) +B ➔ WRONG

OR

(S + B) +A ➔ WRONG

OR

S + (A,B) ➔ WRONG

This ("joint") weaving will be discussed later

Interference Example

General description:

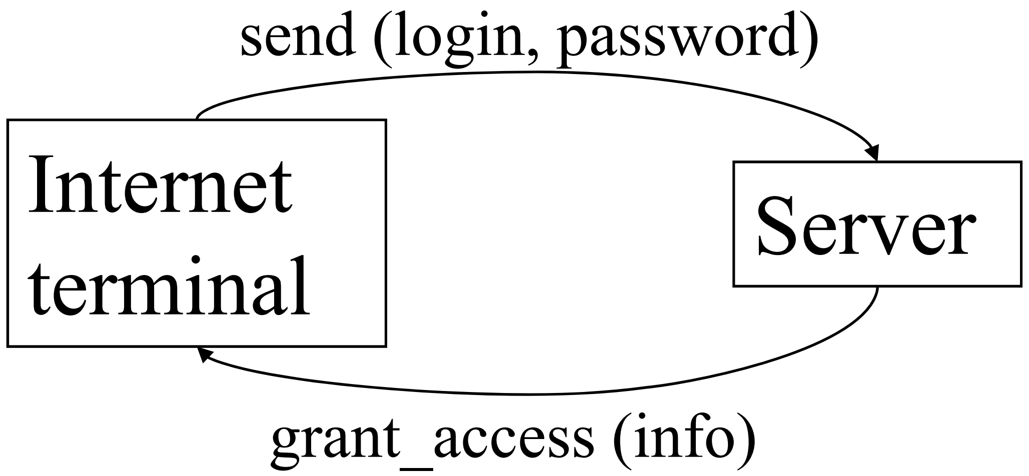
- Two aspects – part of a security-aspects library, to be used in password-protected systems
- **Aspect E** encrypts passwords

  Whenever a password is sent from the login screen of the system, it is encrypted (there is also a decryption part, but we ignore it here)
- **Aspect F** for retrieving forgotten passwords

  Adds a button to report that the password is forgotten. When the button is pressed, security questions are asked. If the answers are correct, the password is sent to the user.

Example Usage: Internet Access to Bank Accounts

Underlying system:



Adding Password Encryption

Aspect E, responsible for encryption.

E's pointcut: a password is sent from login screen

E's assumption, P_E: password-containing messages are sent only from login screen

E's guarantee, R_E: each time a password is sent, it is encrypted

Later addition: aspect F

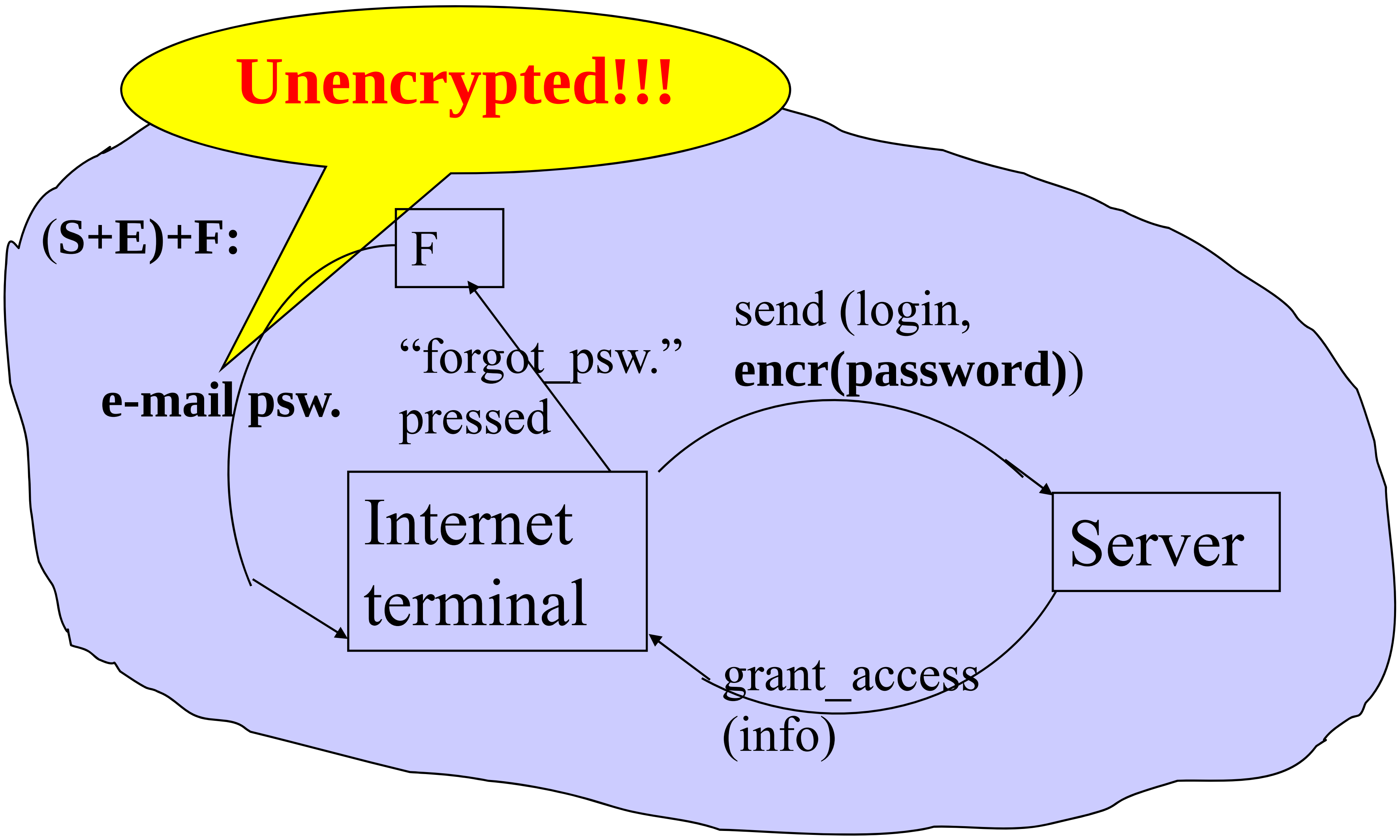
Aspect F, retrieving forgotten passwords:

F’s pointcut: “forgot_password” button is pressed

F’s assumption, P_F: *true* (no assumption needed)

F’s guarantee, R_F: each time a password is forgotten, it’s e-mailed to the user, provided the security questions are answered

Example – contd.(3)



Cause of the problem

- Common join-points? – No.
- Updating shared variables? – No.
- Changing join-points? – Not as written.
- The semantics of E and F? – **Yes!**

1. The presence of **F** (resulting in e-mailed passwords) **violates the guarantee of E** (all passwords encrypted) ➔ F cannot be woven after E.
2. The presence of **F** (e-mailed passwords) **violates the assumption of E** (passwords sent from Login Screen only) ➔ E cannot be woven after F

Semantic Interference – more formally

A – aspect, specified by **(P_A, R_A)**

B – aspect, specified by **(P_B, R_B)**

We assume both aspects are correct

Definition: A does not interfere with B if for every system S,

$$(S \models P_A \wedge P_B) \rightarrow ((S + A) + B \models R_A \wedge R_B) \qquad (*)$$

both assumptions hold

both guarantees hold

(*) Notation: $\mathbf{OK_{AB}}$

Non-Interference in a Library

- Generalization of the definition to a library of N aspects:

The aspect library is interference free if for every subset of the aspects, when they are woven into a system that satisfies all their assumptions, the resulting system satisfies all the guarantees

- We detect interference or prove interference-freedom using model-checking, where advice is modeled as state-transition system

Proving Non-Interference

- **Need to prove:** OK_{AB} and OK_{BA}
- **Intuitive method:** Direct proof.
- For every system S satisfying $P_A \wedge P_B$,
  show that ((S+A)+B) and ((S+B)+A) satisfy $R_A \wedge R_B$
- But: What about N aspects in a library?
- Pairwise checks are not enough!
  Need to prove for **every subset** of aspects separately!
  (for all the subsets of 2,3,…N aspects)

Incremental Non-Interference Proof

Theorem (dividing the proof task):

To prove $\mathbf{OK_{AB}}$, it's enough to show

$$\forall S((S \models P_A \wedge P_B) \rightarrow (S + A \models P_B)) \quad [\mathbf{KP_{AB}}]$$

A keeps the assumption of **B**

And

$$\forall S((S \models R_A \wedge P_B) \rightarrow (S + B \models R_A)) \quad [\mathbf{KR_{AB}}]$$

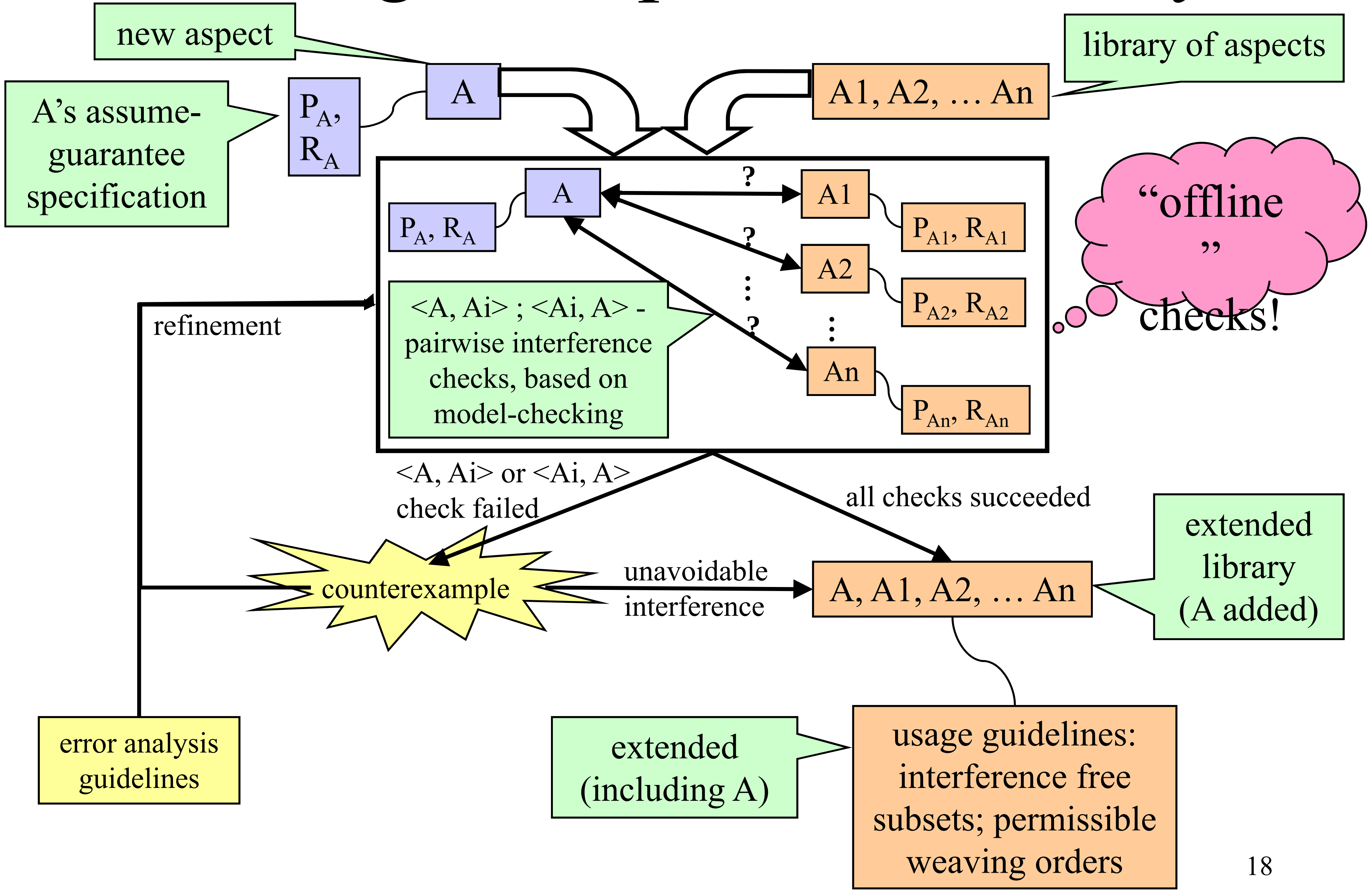
B keeps the guarantee of **A**

The Incremental Method Generalizes to N

- If N aspects pairwise satisfy KP and KR in both directions, for any combination of $m \le N$ aspects from that set, there is no semantic interference.
- Each one preserves the assumption and guarantee of all the others, so no matter how many are applied, all guarantees will hold if all assumptions held in the base
- The above generalization does NOT hold for the Direct method.

example – soon!

Adding an Aspect to a Library


new aspect
A's assume-guarantee specification
P_A, R_A
A
A1, A2, … An
library of aspects
A
P_A, R_A
?
A1
P_{A1}, R_{A1}
?
A2
P_{A2}, R_{A2}
?
An
P_{An}, R_{An}
<A, Ai> ; <Ai, A> - pairwise interference checks, based on model-checking
"offline" checks!
refinement
<A, Ai> or <Ai, A> check failed
all checks succeeded
counterexample
unavoidable interference
A, A1, A2, … An
extended library (A added)
error analysis guidelines
extended (including A)
usage guidelines: interference free subsets; permissible weaving orders

Non-generalization of Direct: Example

- **Aspect A:** Encrypts "secret" data sent in the system
  - In the bank system, encrypts passwords sent from login screen
- **Aspect B:** Adds a possibility to "remember" the password of the user
  - Adds a private variable "password" to the User class, and stores the password there if needed.
- **Aspect C:** "Publishes" data of specified non-secret objects [objects with no "secret" fields] – sends all the object data (including private fields) upon request.
  - In the bank system – sends user data.

Aspect Specifications:

- **Aspect A:**
  - Assumes the password are the only type of secret data, and the passwords are sent only from the login screen
  - Guarantees all the secret data is sent encrypted
- **Aspect B:**
  - Assumes nothing (adds the “save_password” button itself)
  - Guarantees the password is stored in the user data if it was requested
- **Aspect C:**
  - Assumes user objects store no secret data
  - Guarantees all stored user data is sent

Interference?

- **Incremental method:**
  - Verification of KP_{BC} fails
  - Interference among the aspects is detected by pairwise checks alone
- **Direct method:**
  - All pairwise interference checks succeed!
  - But: the aspects do interfere when all three are applied! Aspect C violates the guarantee of A, by sending passwords unencrypted after B saves them.

B violates C’s assumption: password might be “secret”

How??? – C’s assumption is only checked for the original base system, not for the system with B woven

problem!

Feasibility of Composition

≡ non-contradicting specifications

A – aspect, specified by **(P_A, R_A)**

B – aspect, specified by **(P_B, R_B)**

Definition: composition of **A before B** is **feasible** iff all the following formulas are satisfiable:

$P_A \wedge P_B$ (the assumptions are not contradictory)

$R_A \wedge P_B$ (the guarantee of A and the assumption of B)

$R_A \wedge R_B$ (the guarantees are not contradictory)

Feasibility Check

- Recommended to perform in case interference was detected
- Might be performed even before the verification starts, but is not essential
- Is easier and quicker than the full verification process
- If fails – the aspects can not be woven together into a system without changing their specifications (and maybe also their advice)

Automatic and Modular Interference Detection

- Both for **Direct and Incremental** method
- The MAVEN tool – **extended**: improved and adopted for interference-detection purpose
- Original purpose of MAVEN: automatic modular verification of assume-guarantee aspect specifications

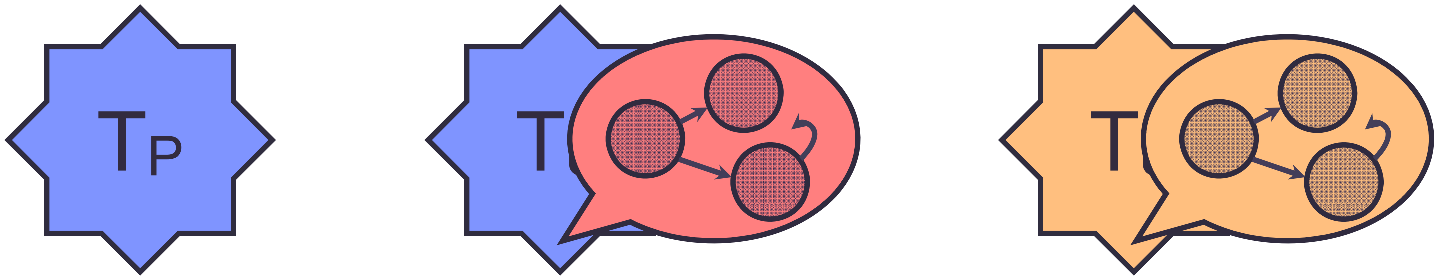
Strategy – MAVEN tool

prior work

- **Build** a "generic" state machine version **(T_P)** of assumption **P_A** (called "tableau")
- **Weave** the aspect **(A)** into this model
- **Prove** that this augmented generic model **(T_P+A)** satisfies the desired result, **R_A**

representation of all the possible systems satisfying P_A

by running NuSMV model-checker



Direct Proof Method

1. Build tableau T for $P_A \wedge P_B$

2. Use MAVEN to prove OK_{AB}

- weave A into T, then weave B
- show $R_A \wedge R_B$ on the result

3. Use MAVEN to prove OK_{BA}

- weave B into T, then weave A
- show $R_A \wedge R_B$ on the result

Incremental Proof Method

Verify $\mathbf{KP_{AB}}$, $\mathbf{KR_{AB}}$, $\mathbf{KP_{BA}}$, $\mathbf{KR_{BA}}$:

1. **Use MAVEN to prove $\mathbf{KP_{AB}}$**
   - build tableau T_P for $P_A \wedge P_B$
   - weave A into T_P
   - show P_B on the result

2. **Use MAVEN to prove $\mathbf{KR_{AB}}$**
   - build tableau T_R for $R_A \wedge P_B$
   - weave B into T_R
   - show R_A on the result

$\rightarrow OK_{AB}$

A maintains the assumption of B

B maintains the guarantee of A

3, 4 (for $\mathbf{KP_{BA}}$, $\mathbf{KR_{BA}}$) – symmetric ($\rightarrow$ **$\mathbf{OK_{BA}}$**)

Incremental method – advantages beyond generalization to N

1. Easier weaving
2. Quicker verification

$\left.\begin{array}{l}\\ \\ \end{array}\right\}$ Cause: smaller models and TL formulas => lower complexity

3. Incremental verification during library construction, and not when a system is run:
   When adding an aspect to the library, allows checking only the new aspect vs. all the rest
4. Advantage in failure analysis:

Depending on the verification step at which we obtained the counterexample, we will know exactly which aspect caused interference and how (= which property was violated)

Error Analysis

- Who is guilty (failure localization), and what is to be done (failure treatment)?
- Failure localization:

  Which assertion was violated?

  Which aspect is responsible for the failure?
- Failure treatment:

  Should the specification of any aspects be changed?

  Should some advice be changed?

Failure Localization

- In Direct method – problematic.
- In Incremental method – straightforward:
  - Immediately follows from the verification stage that failed :

    KP_{AB} failed => A’s advice violates B’s assumption.

    KR_{AB} failed => B’s advice violates A’s guarantee
  - Possible to detect and localize multiple failures (i.e., when both properties are violated)

Failure Treatment

- Feasibility check fails =>
  - Specifications **have** to be changed
  - Advice implementation **might** have to be changed
- Feasibility check succeeds =>
  - Advice implementation **has** to be changed
  - Specifications **might** have to be changed
- Failure elimination impossible =>

  Usage guidelines for the aspects (restrictions on the possible weaving order)

Bank System Example - Reminder

S: system providing internet access to bank accounts. Involves sending passwords from “login” screen

E: aspect in charge of encrypting the passwords sent from login screens

F: aspect in charge of retrieving forgotten passwords; sends them by e-mail

Bank system – Verification Failures

- $\mathbf{KR_{EF}}$ fails ➔ F can not be woven after E, because it does not preserve the guarantee of E, $\mathbf{R_E}$ (the e-mailed password will be unencrypted)
- $\mathbf{KP_{FE}}$ fails ➔ F can not be woven before E, because F violates the assumption of E, $\mathbf{P_E}$ (the passwords are sent not only from the “login” screen)

Bank system – Error Analysis

- Example: KP_{FE} check failed, but
- Feasibility check succeeds
- Possible solution: Change the advice of F!
  - For example:

    Change F to bring the user to a login screen and offer to enter the new password
  - Result: Specifications stay the same, but OK_{FE} now holds, so we can weave F before E (but not the reverse)

Joint Weaving

- At every point of the program decides which of the aspects to apply and in which order
- When is **joint weaving** equivalent to **sequential**?
  - (S + (A,B)) ≡? ((S+A)+B)
  - (S + (A,B)) ≡? ((S+B)+A)

Joint Vs. Sequential Weaving - 1

Notation: $J_A(S)$ = set of join-points of A in S

If:

A and B have no common join-points

B does not affect the set of A's join-points

- $J_A(S) \cap J_B(S) = \varnothing$
- $J_A(S+B) = J_A(S)$ =>
- $J_B(S+A) = J_B(S)$

A does not affect the set of B's join-points

Then:

$(S + (A,B)) \equiv ((S+A)+B) \equiv ((S+B)+A)$

Both orders of sequential weaving are equivalent to the joint weaving

Joint Vs. Sequential Weaving - 2

A and B have no common join-points

B does not affect the set of A's join-points

If:

- $J_A(S) \cap J_B(S) = \emptyset$
- $J_A(S+B) = J_A(S)$
- $J_B(S) \subseteq J_B(S+A) \subseteq J_B(S) \cup S_A$

$\}$ =>

A does not remove join-points matched by B

A might add join-points matched by B, but only inside A's advice

Then:

$(S + (A,B)) \equiv ((S+A)+B)$

Joint weaving of A and B is equivalent to first weaving A and then B

Interference Detection in Java Systems

- Work in progress : industrial case study

  Toll System (Siemens) – charging for road use
  - Formalization of aspect specifications
  - Translating advice to transition systems
  - Verification of aspects and interference detection

  Intermediate results:
  - Interference between two aspects found and is being analyzed now

Interference Detection in Java Systems(2)

- Planned: case study based on **library of reusable aspects** that implement ACID properties for transactional objects

atomicity
consistency
isolation
durability

- Large library of aspects, intended to be used as benchmark
- Authors state there is interference between the aspects
- Goal: formalization, analysis => interference warnings and non-interference proofs for the aspects => usage guidance for the library

More Work in Progress

- Generalizing the proof method
  - More weaving strategies
  - Extending MAVEN
- Refining the error analysis
- Running more complicated examples
- The formalization and proof method can be extended to treat other types of aspect interactions, such as cooperation [one aspect establishes the assumption of another…]

Summary

new! Semantic interference among aspects is defined

new! Interference-detection method is modular and incremental

new! Verification result is not "yes" or "no"! The method gives usage guidelines for the library

- For any comments / questions, please write to {emika,katz}@cs.technion.ac.il

Thank you!