Modular Generic Verification of LTL Properties for Aspects

Shmuel Katz
Max Goldman

{katz, mgoldman}@cs.technion.ac.il

FOAL ’06
Aspects

Base + Aspect = Augmented

State Machines

Model Checking
Aspect Verification

Aspects have a specification

- Requirements about base system
- Results to hold in augmented system

Prove once-and-for-all that an aspect satisfies its specification
Aspect Verification

Aspect requires blue and guarantees orange.
Modular

Consider the aspect independently from the base machine

Prove guarantees

Prove is
Consider the aspect independently from any base machine.

Prove guarantees or is
Idea
Aspect

Advice: state machine A

Pointcut: descriptor \( \rho \)

Specification:

- Base machine requirement \( \psi \)
- Woven machine result \( \varphi \)
Aspect

Advice: state machine

Pointcut: descriptor $\rho$

Specification:

- Base machine requirement $\psi$
- Woven machine result $\varphi$
Aspect

Advice: state machine A

Pointcut: descriptor $\rho$

Specification:

- Base machine requirement $\psi$
- Woven machine result $\varphi$
Aspect

Advice: state machine A

Pointcut: descriptor $\rho$

Specification:
  - Base machine requirement
  - Woven machine result $\varphi$
Aspect

Advice: state machine A

Pointcut: descriptor ρ

Specification:

- Base machine requirement ψ
- Woven machine result φ
Aspect

- Advice: state machine A
- Pointcut: descriptor $\rho$
- Specification:
  - Base machine requirement $\psi$
  - Woven machine result
Aspect

Advice: state machine A

Pointcut: descriptor $\rho$

Specification:

- Base machine requirement $\psi$
- Woven machine result $\varphi$
Goal

Prove

— For all base machines B

— If “B satisfies $\psi$”

— Then “B woven with A according to $\rho$ satisfies $\varphi$”
Problem

What if the aspect puts the base program into a state it could never reach on its own?

The behavior of the base program is unknown
Weakly Invasive

Aspect returns to the base program only in states reachable by that base program on its own

— Spectative
— Regulative
— Invasive within original domain
Prove

- For all base machines $B$
- If “$B$ satisfies $\psi$”
- And “$A$ with $\rho$ is weakly invasive for $B$”
- Then “$B$ woven with $A$ according to $\rho$ satisfies $\varphi$”
Strategy

- Build a “generic” state machine version of assumption $\psi$
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result
Strategy

- Build a “generic” state machine version of assumption $\psi$
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result
Strategy

- Build a “generic” state machine version of assumption $\psi$
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result

$T_\psi$ $T$
Strategy

- Build a “generic” state machine version of assumption $\psi$
- Weave the aspect into this model
- Prove that this augmented generic model satisfies the desired result
Components
State Machines

- Finite set of states
- Set of atomic propositions
- Labels
- Nondeterminism
State Machines

- Finite set of states $S$
- Set of atomic propositions $AP$
- Labeling function $L : S \rightarrow 2^{AP}$
- Path relation $R$ containing pairs $(s,t)$ when there is a transition from $s$ to $t$
State Machine
State Machine
State Machine

State Diagram:

- State: abc
- State: b
- State: c
- State: ab
- State: ac
- State: bc
- State: abc
Fairness

Problem with nondeterminism: often allows the system to “do nothing” forever

Impose a fairness constraint, and only look at fair paths

Fairness set $F$: set of subsets of $S$

A path is fair iff it visits every set in $F$ infinitely often
Fairness
Fairness
Fairness
Fairness

- $a b c$
- $b$
- $c$
- $a b c$
- $b$
- $c$
- $a c$
- $b$

Diagram:
Fairness

\begin{align*}
\begin{array}{c}
\text{abc} \rightarrow \text{b} \\
\text{abc} \rightarrow \text{c}
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c}
\text{b} \rightarrow \text{c} \\
\text{b} \rightarrow \text{a}
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c}
\text{abc} \rightarrow \text{a}
\end{array}
\end{align*}
Linear Temporal Logic

Logic of infinite paths of computation

Path formulas

\[ G \, p \]

\[ p \rightarrow F \, (q \, U \, r) \]

\[ p \rightarrow X \, q \]
LTL Formulas

- $\mathcal{A} \mathcal{F} c$
- $\mathcal{A} \mathcal{F} \mathcal{G} \neg b$
- $\mathcal{A} \mathcal{G} (\neg a \land b) \rightarrow \mathcal{F} a$
Base Machine

State machine B

Computation starts from one of the initial states $S_0 \subseteq S$
Base Machine

\[ S_0 \]

Diagram of a base machine with states labeled as 'a b c', 'b', and 'c'.
Advice

State machine A

- Initial states $S_0$
- Return states $S_{ret}$
Advice

\[ S_0 \quad \text{a b} \quad \text{S}_{\text{ret}} \quad \text{b} \]
Pointcuts

Pointcut descriptor $\rho$

- Matches the end of a path
- Past LTL, regular expressions, ...
Pointcut

\[
\rho = a \land Y b \land Y Y b
\]
Components

- State machines
- Fairness
- LTL
- Base machines
- Aspect advice machines
- Aspect pointcuts
Weaving

Inputs:

— Base machine B
— Aspect machine A
— Pointcut $\rho$

Output:

— Woven machine $\tilde{B}$
Weaving A with B

---

Step 1: Make B pointcut-ready for $\rho$

- Result: Machine $B^\rho$

---

Step 2: Augment $B^\rho$ with A

- Result: Augmented machine $\breve{B}$
1. Pointcut–Ready

- Advantage: simplicity
- Disadvantage: static, not dynamic
- No problems for many aspects
  - State pointcut
  - Method call pointcut
1. Pointcut-Ready

Unwinding of paths such that each state either definitely does or definitely does not match the pointcut

Matching states are labeled ‘pointcut’
1. Pointcut-Ready

\[ \rho = a \land Y b \land Y Y b \]
1. Pointcut–Ready

\[ \rho = a \land Y b \land Y Y b \]
1. Pointcut-Ready

\[ \rho = a \wedge Y b \wedge Y Y b \]
2. Augmented

Transitions from base machine ‘pointcut’ states to aspect initial states

Transitions from aspect return states to base machine states

According to state labels
2. Augmented

Rule: add all edges

- ‘pointcut’ → aspect initial
- aspect return → base

Where the labels match
Weakly Invasive

All edges from aspect return states go to reachable states in the base machine
Tableaux
Recall

A “generic” model built from the assumption formula $\psi$

$T_\psi$
Tableaux

Exactly all the paths which satisfy a given LTL path formula
Tableau
Tableau

F b
Tableaux

For a given LTL formula $\psi$

- If a path supports the formula, it must be in the tableau

For any machine satisfying $\psi$

- All its paths must be in the tableau
Algorithm
Recall

- Advice: state machine A
- Pointcut: descriptor $\rho$
- Specification:
  - Base machine requirement $\psi$
  - Woven machine result $\varphi$
- $A, \rho, \psi, \text{ and } \varphi$ over AP
Step 0

Throw all the atomic propositions in AP into \( \psi \), in clauses of the form

\[ \cdots \land (a \lor \neg a) \]
Step 1

Construct $T_\psi$, the tableau for $\psi$
Step 2

Restrict $T_\Psi$ to its reachable component
Step 3

Weave A into $T_\psi$ according to $\rho$

Result: $\tilde{T}_\psi$
Step 4

Determine if $\tilde{T}_\psi \models \varphi$
Claim

If $\tilde{T}_\psi \models \varphi$

Then for any $M$

- If $M \models \psi$

- And $A$ and $\rho$ are weakly invasive for $M$

- Then $\tilde{M} \models \varphi$
Proof
Outline

- $T_\psi$ has every possible path
- So $\tilde{T}_\psi$ has every possible augmented path
- If $\tilde{T}_\psi \models \varphi$
- Then every possible augmented path supports $\varphi$
Example
Aspect

\[ \psi = A \ G ((\neg a \land b) \rightarrow F a) \]

\[ \varphi = A \ G ((a \land b) \rightarrow X F a) \]

\[ \rho = a \land b \]

\[ A = \begin{array}{c}
& a \ b \\
\rightarrow & & \rightarrow \\
& b
\end{array} \]
\widetilde{T} \models \varphi
Result

The aspect satisfies its specification
Really?

Diagram:

- Node labeled 'abc' with arrows pointing to nodes labeled 'b' and 'c'.
- Node labeled 'b' with an arrow pointing to a node labeled 'ac'.
- Node labeled 'c' with an arrow pointing to a node labeled 'b'.
Really.
Aspect Verification

- Prove once-and-for-all that an aspect satisfies its specification
- Modular
- Generic
- Uses an LTL tableau as a “generic” model
- More on the way
Modular Generic Verification of LTL Properties for Aspects

Shmuel Katz
Max Goldman

{katz, mgoldman}@cs.technion.ac.il

FOAL ’06