Modular Verification of Strongly Invasive Aspects

Authors:
Emilia Katz, Shmuel Katz
{emika,katz}@cs.technion.ac.il

The Technion

Modular verification of aspects - Motivation

- Enables reuse without proof:
  - An aspect that is proven to be “correct” can be woven into any “suitable” base system, without additional checks
  - If several “correct” aspects have the same requirements from the base system, and it satisfies these requirements, each of the aspects can be applied to it without further checks (one at a time)
- Leads to smaller models

  => Model-checking is enhanced, and sometimes even made possible

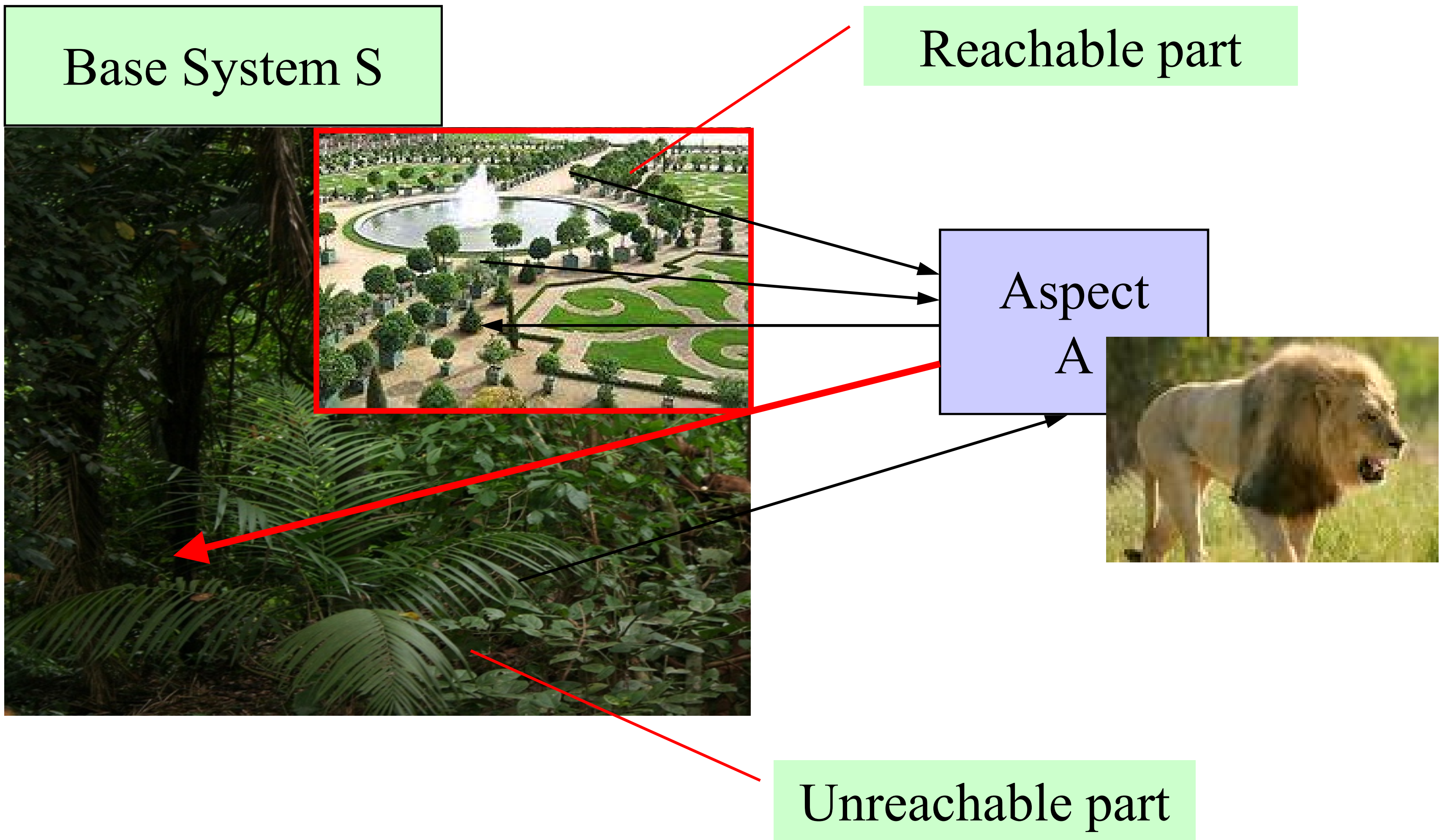
The Setting: Aspect Representation

- Advice = state machine
  - Abstract representation as state-transition system
  - Obtained during the modeling stage, or built from code (e.g., by tools like Bandera)
- Pointcut = state predicate about the base system
  - assume that the system has been preprocessed and the join-points states have already been marked
- Weaving (abstract version):
  - Every join-point in the base is connected to the corresponding initial states of the advice (instead of its former next states)
  - Every last state of the advice is connected to all the corresponding states in the base system model

Strongly Invasive Aspects

- All the aspects can be divided into categories, according to their influence on the base system:
  - spectative, regulative, weakly invasive and strongly invasive
  - each category is contained in all the following ones
- Modular verification exists for weakly invasive aspects.
  - Aspects that can only gather information, or change paths in the reachable part of S
- We propose a modular verification technique that works for strongly invasive aspects as well
- What are strongly invasive aspects?
  - Aspects allowed to perform arbitrary modifications to the base system computations
  - Can reach previously unreachable parts of S and thus violate state invariants of the base system

Strongly Invasive Aspects - Intuition



Strongly Invasive Aspect - Example

Aspect B (for “Bonus”):

- To be used in grades-managing systems
- Provides a way of giving bonus points for assignments / exams, including grades above 100
- Still keeps the final grade in 0..100 range

Example – contd.

B's behavior - two kinds of actions:

1. Pointcut: Assignment or exam grade is entered

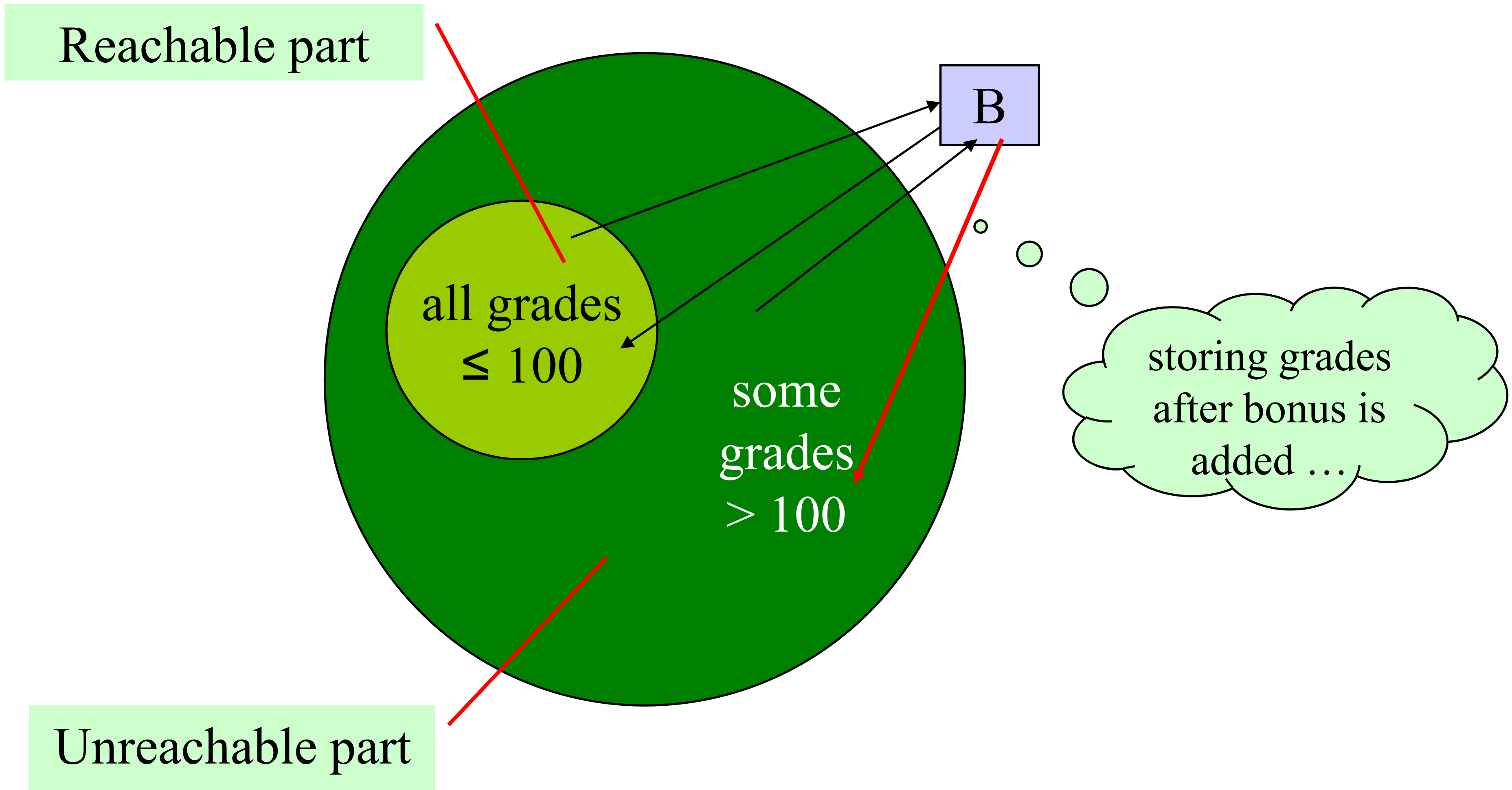
Advice: - offer a possibility of giving a bonus

- store the new grade successfully even if it exceeds 100

2. Pointcut: Final grade calculation of the base system is performed

Advice: if the calculation resulted in a grade that exceeds 100, the aspect replaces this grade by 100 (otherwise keeping the grade unchanged)

Why is B Strongly Invasive?



Why is B Interesting?

- After weaving, the calculations are performed partly in the aspect, and partly in the base system code, but using new, previously impossible, inputs
- Highly reusable: doesn't restrict the grade calculation process of the base system, as long as it can handle values>100
- Can appear in a library of aspects providing different grading policies. Then:
  - All these aspects will have the same assumptions as B, so
  - Enough to check a given base system for applicability of one of the aspects from this library, and applicability of all the others will follow
  - The grading policy can be changed as needed at any time, by replacing the applied aspect, without any further checks on the base system

Refined Aspect Specification

What is a "correct" aspect?

LTL formulas

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

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

A assumes:

in **any reasonable** base system for A

P_A holds in the base system:

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

new!

U_A holds in the unreachable part of the base system:

- what's true for computations starting from all A's resumption states that were unreachable in the base system

A guarantees: R_A is true in the woven system

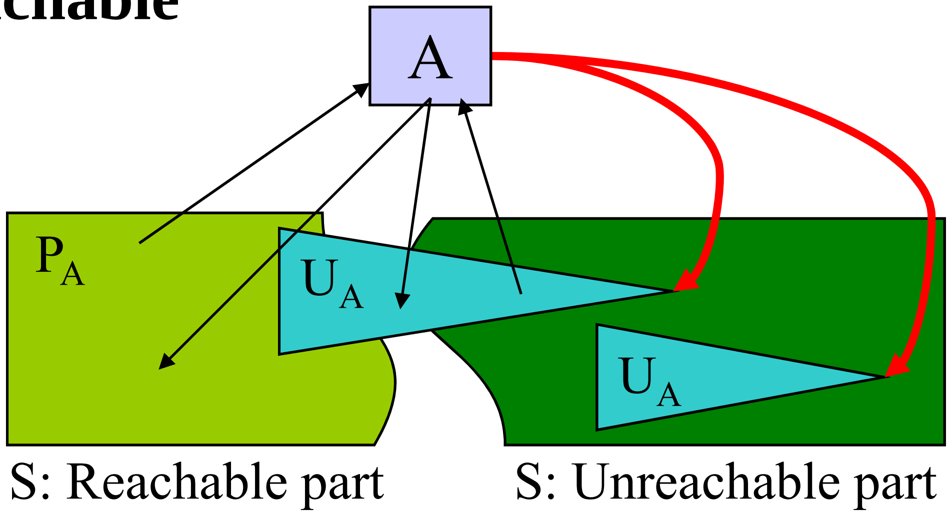
in **any** woven system with A

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

possibly global!

Refined Aspect Specification – contd.

P_A: assumption on reachable
U_A: assumption on unreachable
R_A: guarantee on woven



Example – Aspect B specification

$\mathbf{P_B}$ (B's assumption on the reachable part):

1. All the grades appearing in the grading system are in 0..100 - homeworks (*hw_i*), exams (*exam_j*), final (*f*)
2. After the final grade is ready (*f_ready*) (i.e., all the assignments and exams that comprise the grade have been checked, and the final grade has been calculated from them according to the base system grading policy), the final grade is published (*f_published*).
3. *calc* represents the "ideal" result of the final grade calculation, according to the base system grading policy

$$P_B = [\, G(f\_ready \rightarrow ((f = calc) \land F\ f\_published)) \quad (2)$$

$$G(f\_published \rightarrow f = calc) \land \quad (3)$$

$$\left.\begin{array}{c} G(0 \le f \le 100) \land \\ G(\forall 1 \le i \le 10\ (0 \le hw\_i \le 100)) \land \\ G(\forall 1 \le j \le 2\ (0 \le exam\_j \le 100))\,] \end{array}\right\} (1)$$

Aspect B specification – contd.

$\mathbf{U_B}$ (B's assumption on the unreachable part):

- A weakening of P_B
- All the grades in the system are now in 0..120

$$
\boldsymbol{U_B} = \left[\begin{array}{c}
G(f\_ready \rightarrow ((f = calc) \wedge \mathrm{F}\, f\_published)) \\
G(f\_published \rightarrow f = calc) \wedge \\
G(0 \le f \le \mathbf{120}) \wedge \\
G(\forall 1 \le i \le 10\ (0 \le hw\_i \le \mathbf{120})) \wedge \\
G(\forall 1 \le j \le 2\ (0 \le exam\_j \le \mathbf{120}))
\end{array}\right]
\begin{array}{l}
\left.\begin{array}{c}\\ \\\end{array}\right\} \text{same as in } P_B \\
\left.\begin{array}{c}\\ \\ \\\end{array}\right\} \text{100 changed to 120}
\end{array}
$$

Aspect B specification – contd.

$\mathbf{R_B}$ (B's guarantee):

- Regardless of the existence of bonuses on the components of the final grade, the final grade will be the correct one, calculated according to the base system grading policy, but rounded down to 100 if needed
- R_B might also include a statement about the bonus policy it enforces, saying that the aspect calculates the bonuses as desired …

$$\boldsymbol{R_B} = [\mathrm{G}(f\_published \rightarrow f = \min(calc,\ 100))]$$

Modular Verification as a Whole

- Verify that the aspect is “correct” w. r. t. its assume-guarantee specification
- Before weaving into a concrete base system, check that the base system satisfies all the assumptions of the aspect

Weakly Invasive Aspect Verification

prior work

Given a weakly invasive aspect A with the specification (P_A, R_A),

- Use MAVEN tool to automatically verify that whenever A is woven into a base system satisfying P_A, the resulting system satisfies R_A
- To weave A into a given base system, S: use model-checker (e.g., NuSMV) to verify that all the computations of S satisfy P_A

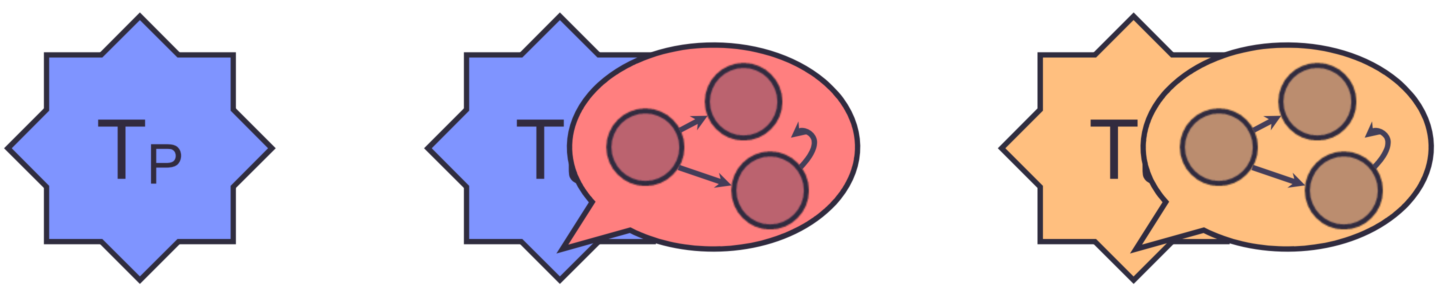
Strategy – MAVEN tool

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**

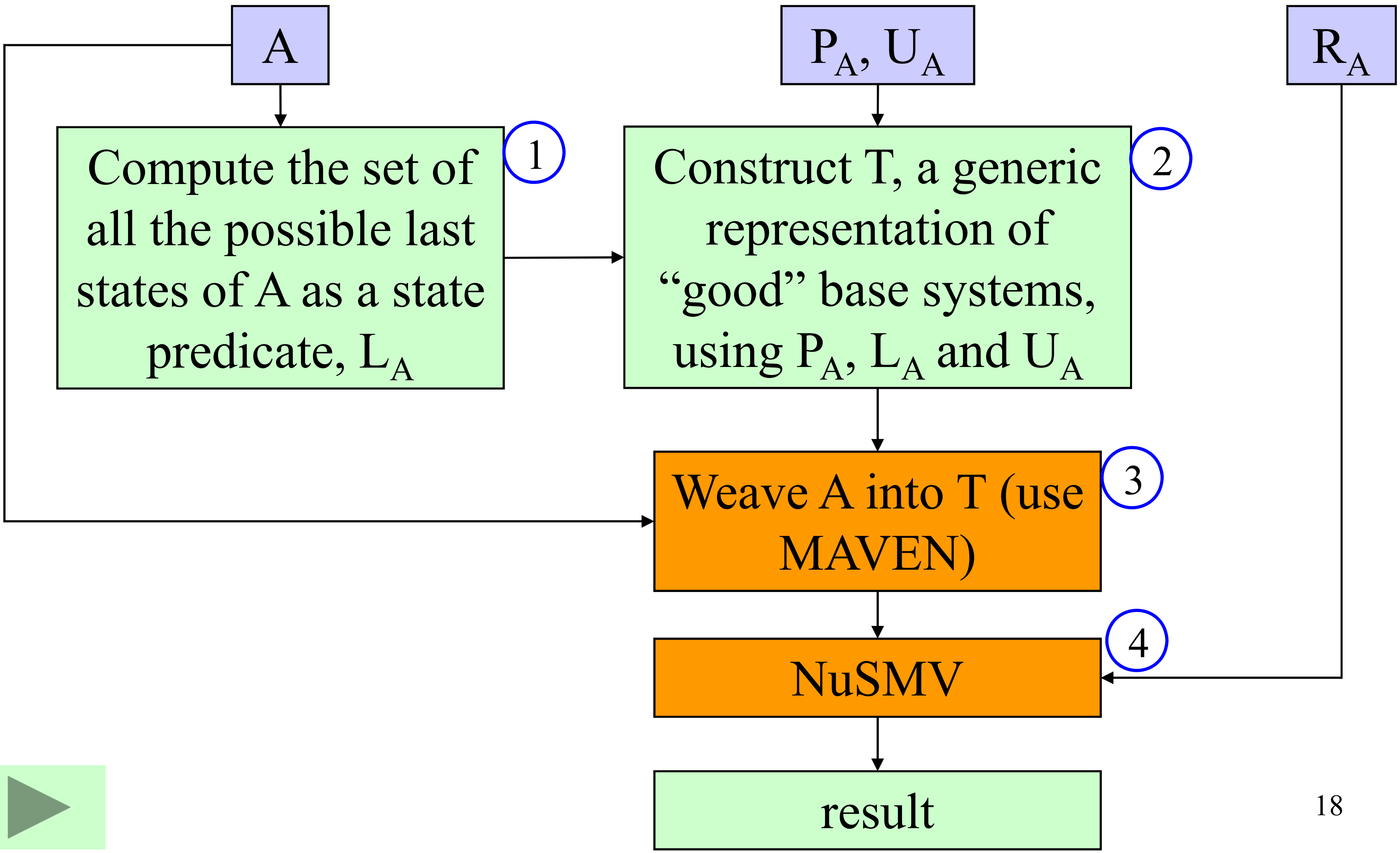
representation of all the possible systems satisfying P_A

by running NuSMV model-checker



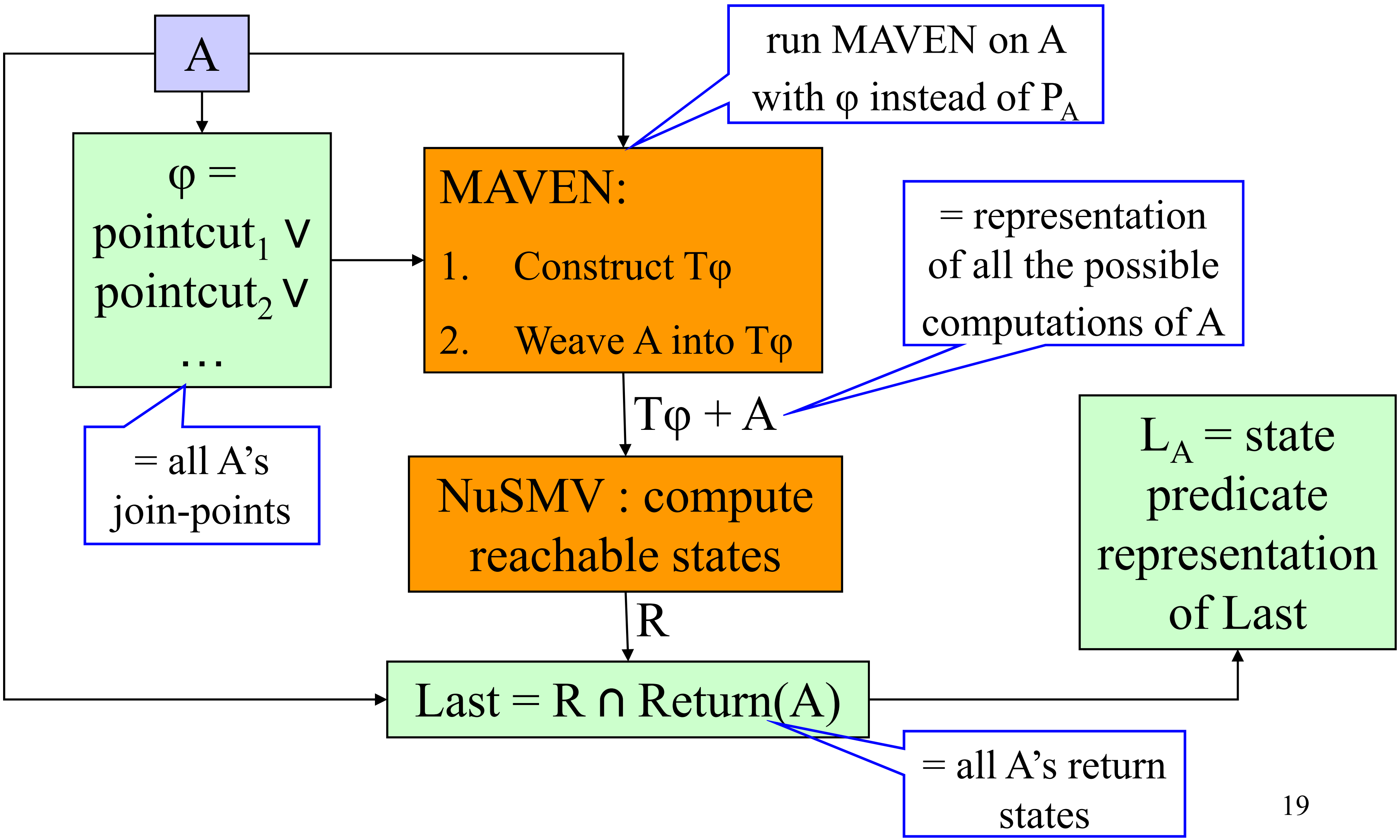
General Aspect Verification – Part 1

(Verifying the Aspect)



General Aspect Verification – Part 1.1

(Computing L_A)



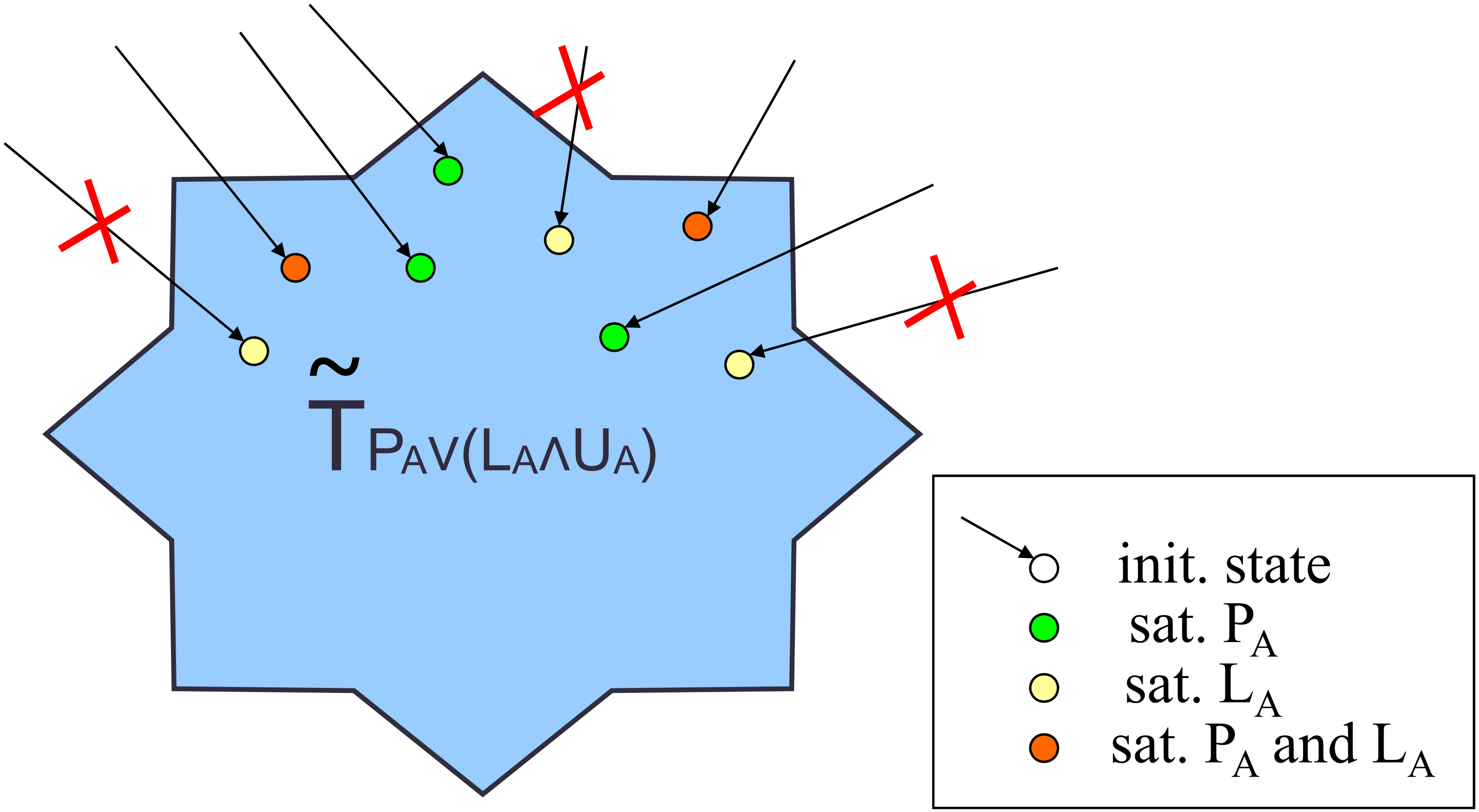
General Aspect Verification – Part 1.1
(Computing L_A) – contd.

- Sometimes it is easy to see a compact description of all the possible last states of A
  - We want to be able to use user-specified predicate L_A in the aspect verification algorithm
  - Need to check the predicates provided
- Checking a user – specified predicate L:
  - Construct the predicate L_A by our algorithm
  - Verify that ($L_A \rightarrow L$) always holds (using a SAT solver)
  - If it does, using L instead of L_A is sound

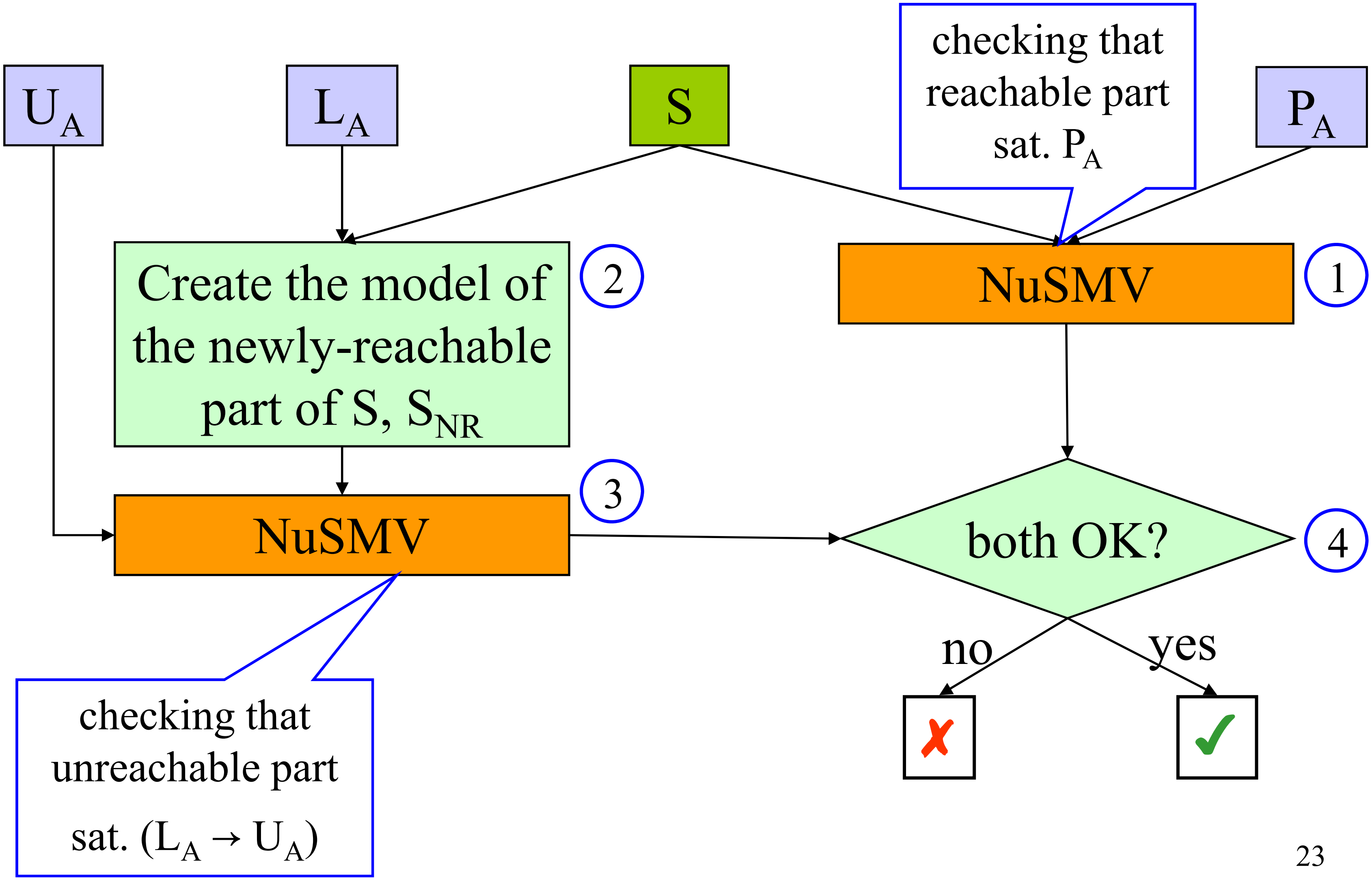
General Aspect Verification – Part 1.2
(Constructing T)

- T should represent all the “good” base systems
- What is a “good” system?
  - The **reachable** part of S satisfies P_A
  - The **unreachable** part of S satisfies $(L_A \rightarrow U_A)$
- What kind of systems do we know how to represent?
  - All the systems the **reachable** part of which satisfies some given LTL formula, φ
  - Can do it automatically, using ltl2smv module of NuSMV
- The idea:
  - “pretend” the interesting part of the unreachable states is reachable
  - construct the representation of such systems
  - correct it to restore the original reachability

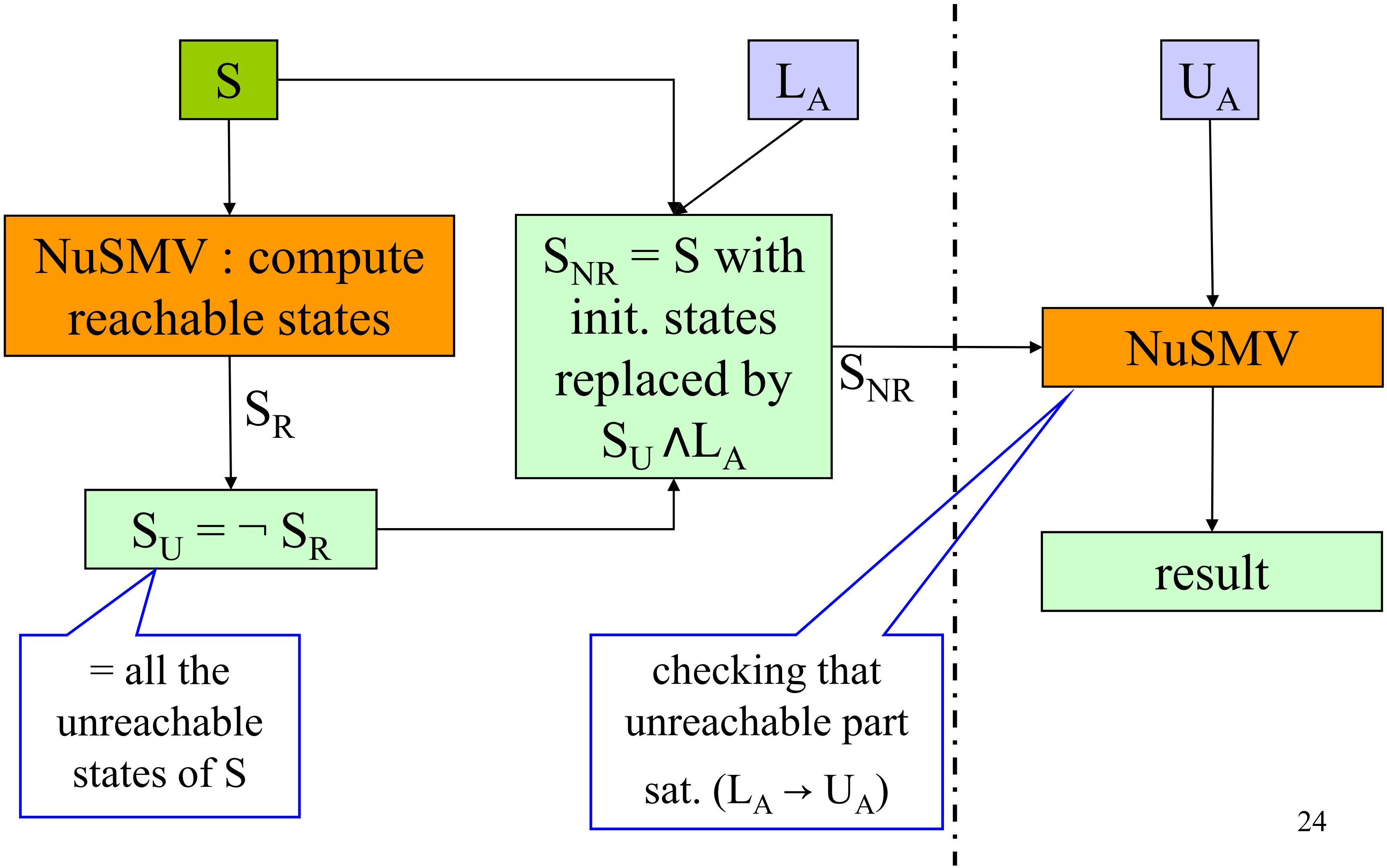
General Aspect Verification – Part 1.2
(Constructing T) – contd.



General Aspect Verification – Part 2
(Checking the Base System)



General Aspect Verification – Part 2.2
(Constructing S_{NR} from the base system)

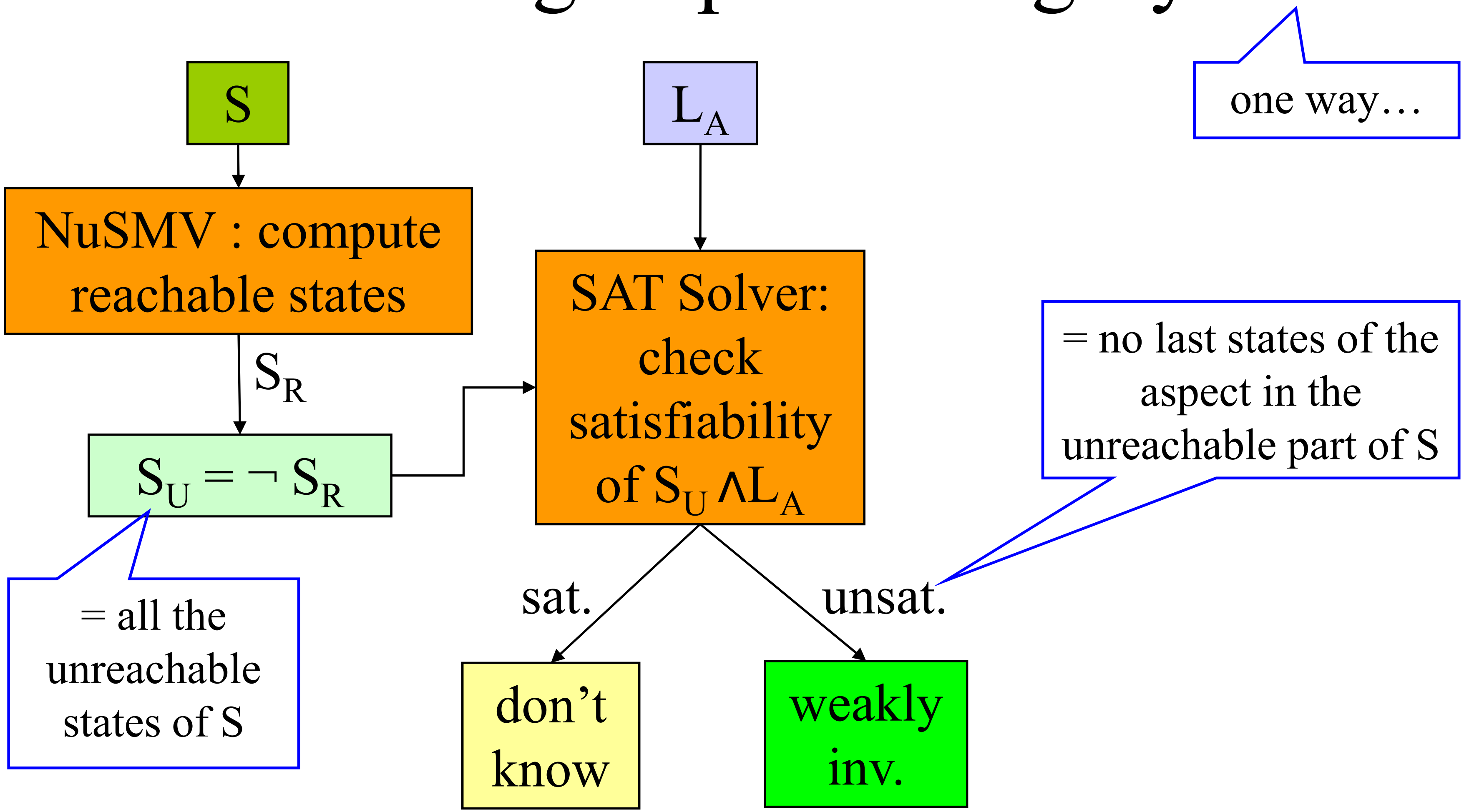


Optimizations

Two places for optimization:

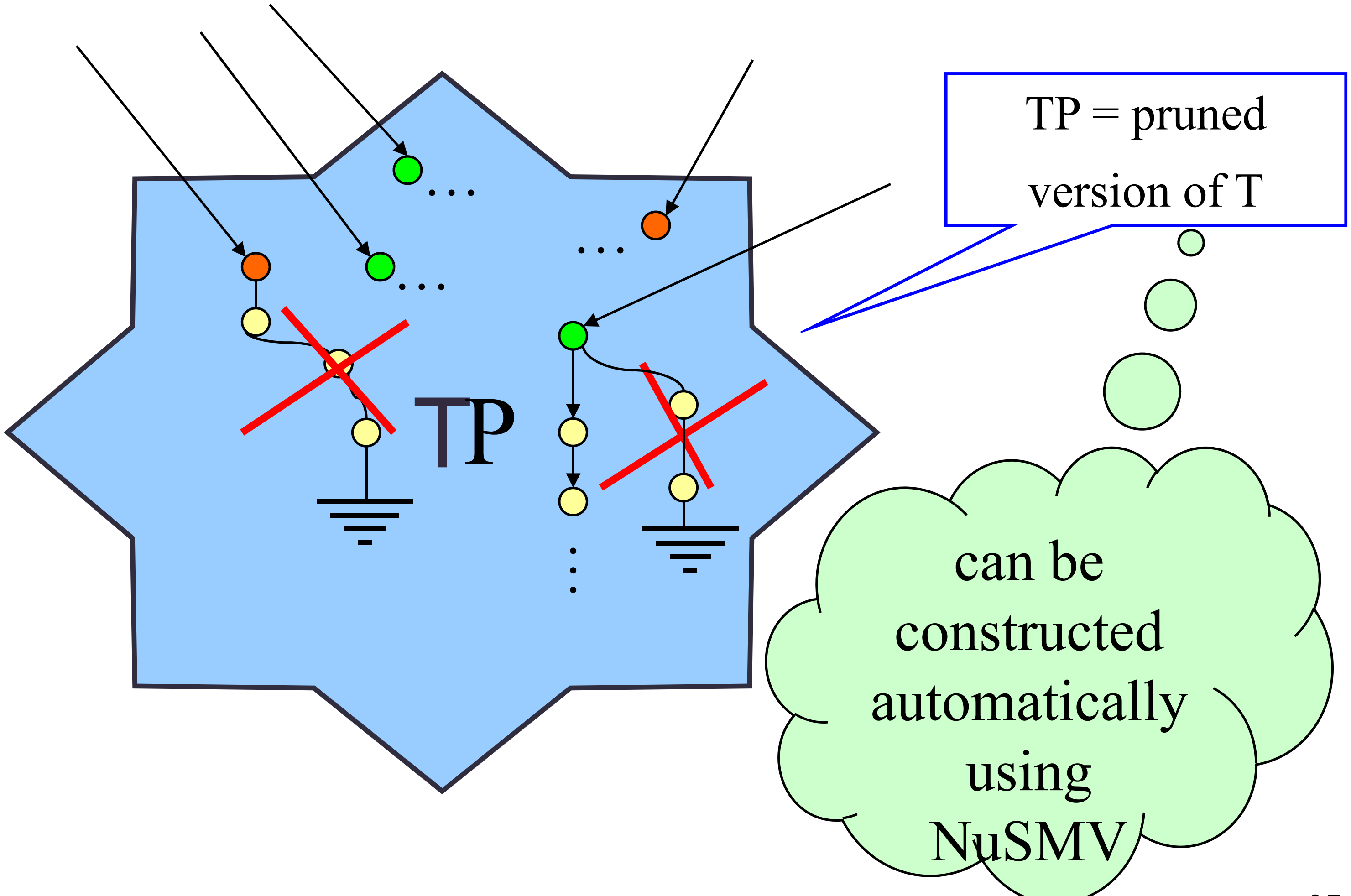
- When the verified aspect is proven to be weakly invasive, a simpler verification method can be used
  - Thus we'd like to be able to check whether a given aspect is weakly invasive
- When base system verification is performed, the requirement on the unreachable part can sometimes be relaxed due to the structure of U_A
  - Then the proof is easier for the model-checker

Determining Aspect Category – 1

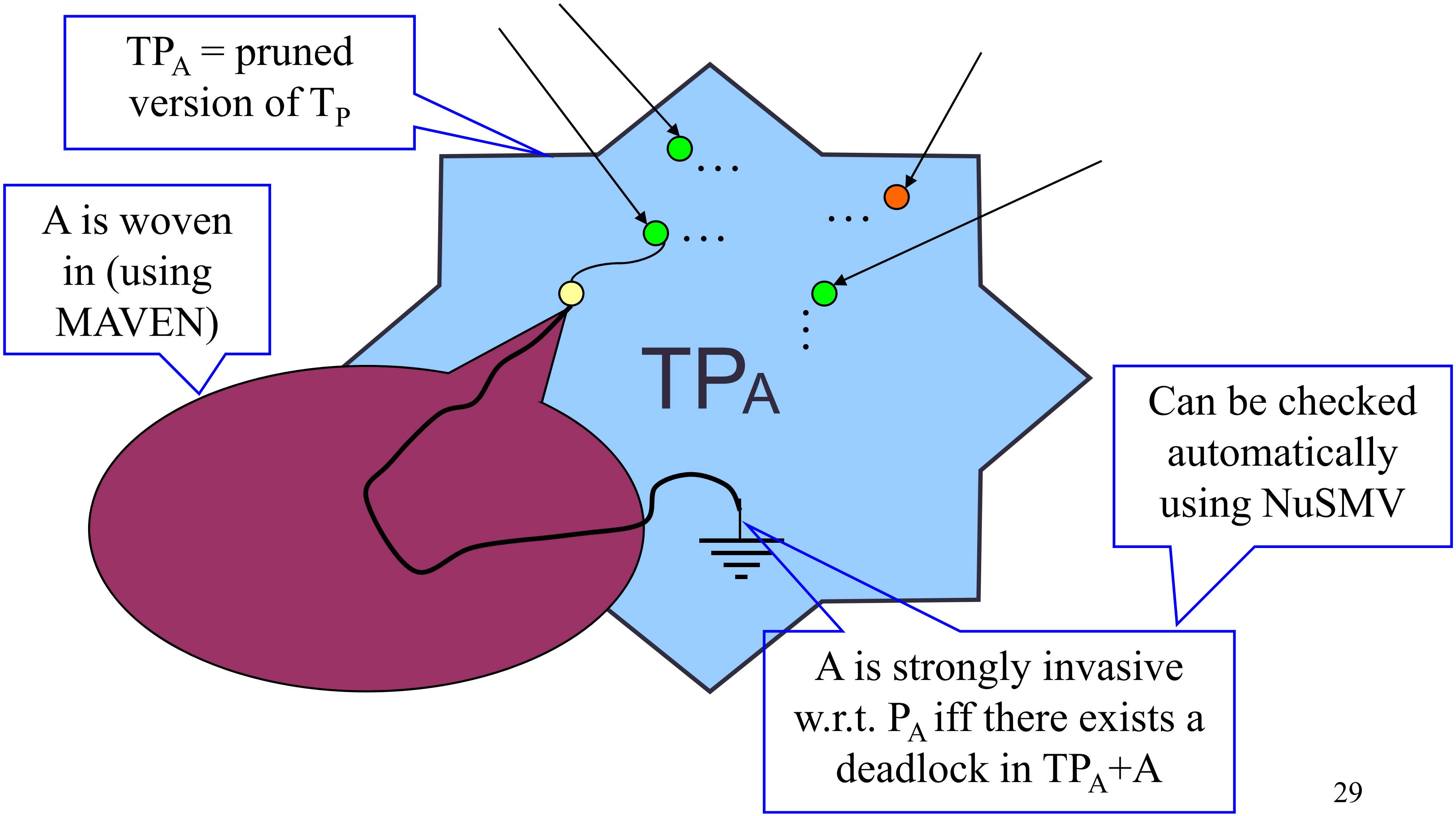


Determining Aspect Category – 2: 1.1. Pruned Tableau

another way…

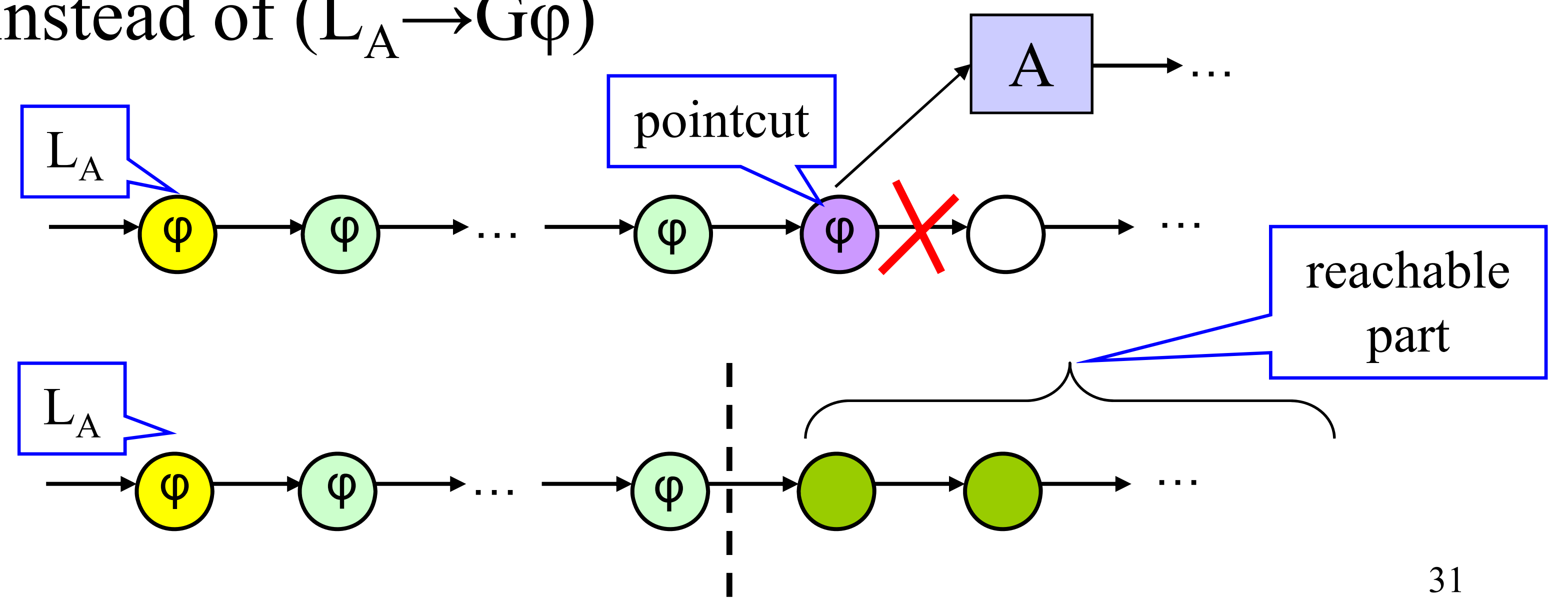


Determining Aspect Category – 2: 1.3. Is A Strongly Invasive w.r.t. P_A?



Optimizing Base System Verification

- If U_A is a safety property ($U_A = G\varphi$):
  - Enough to check φ only in segments between a resumption state of A and the next join-point or reachable state
  - Verify $L_A \rightarrow (\varphi \; \mathsf{U} \; (\text{reachable} \lor (\text{pointcut} \land \varphi)))$ instead of $(L_A \rightarrow G\varphi)$



Summary

new!

- Specification for strongly invasive aspects

new!

- Modular verification method treating aspects of all the categories
- Advantage of modular verification method
  - possibility of reuse without proof:
    - Many base systems satisfying the same assumptions (=> can apply same aspect to many base systems)
    - Many aspects have the same assumptions
      (=> can apply each of the aspects to the same base)

Thank you!