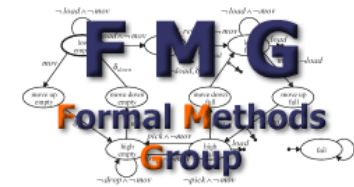

Automatic Generation of Verification Properties for SoC Design from SysML-Diagrams

University of Tuebingen

Computer Engineering Department

Formal Methods Group



Stefan Laemmermann, Roland J. Weiss, Juergen Ruf,
Thomas Kropf, Wolfgang Rosenstiel

{laemmerm,weissr,ruf,kropf,rosen}@informatik.uni-tuebingen.de

**3rd International UML for SoC Design Workshop at
DAC'06**

San Francisco - California, USA

23th July 2006



Outline

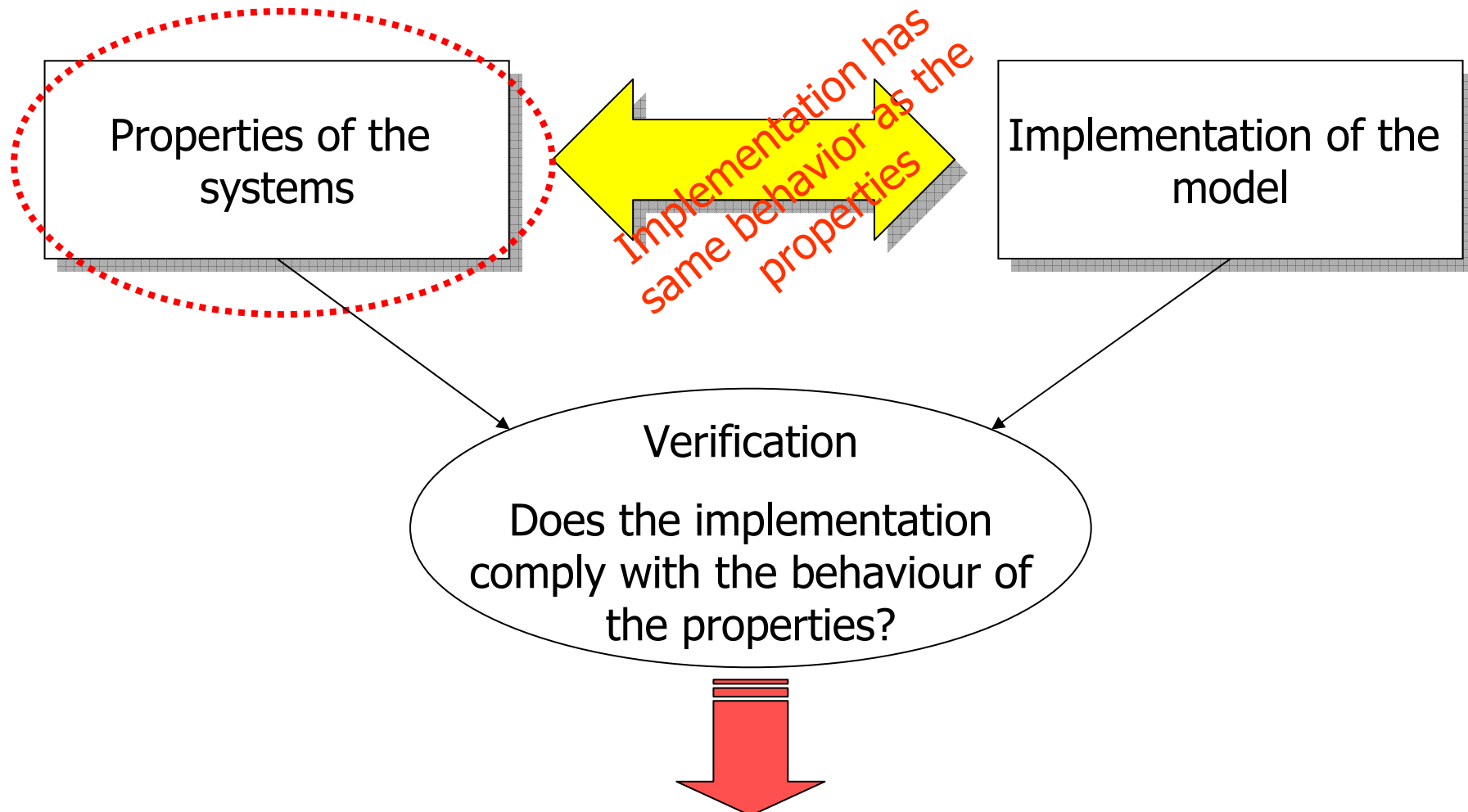
- Motivation and Introduction
- Extraction of Properties
- Applications of the Method
- Conclusion and Future Work



Motivation and Introduction



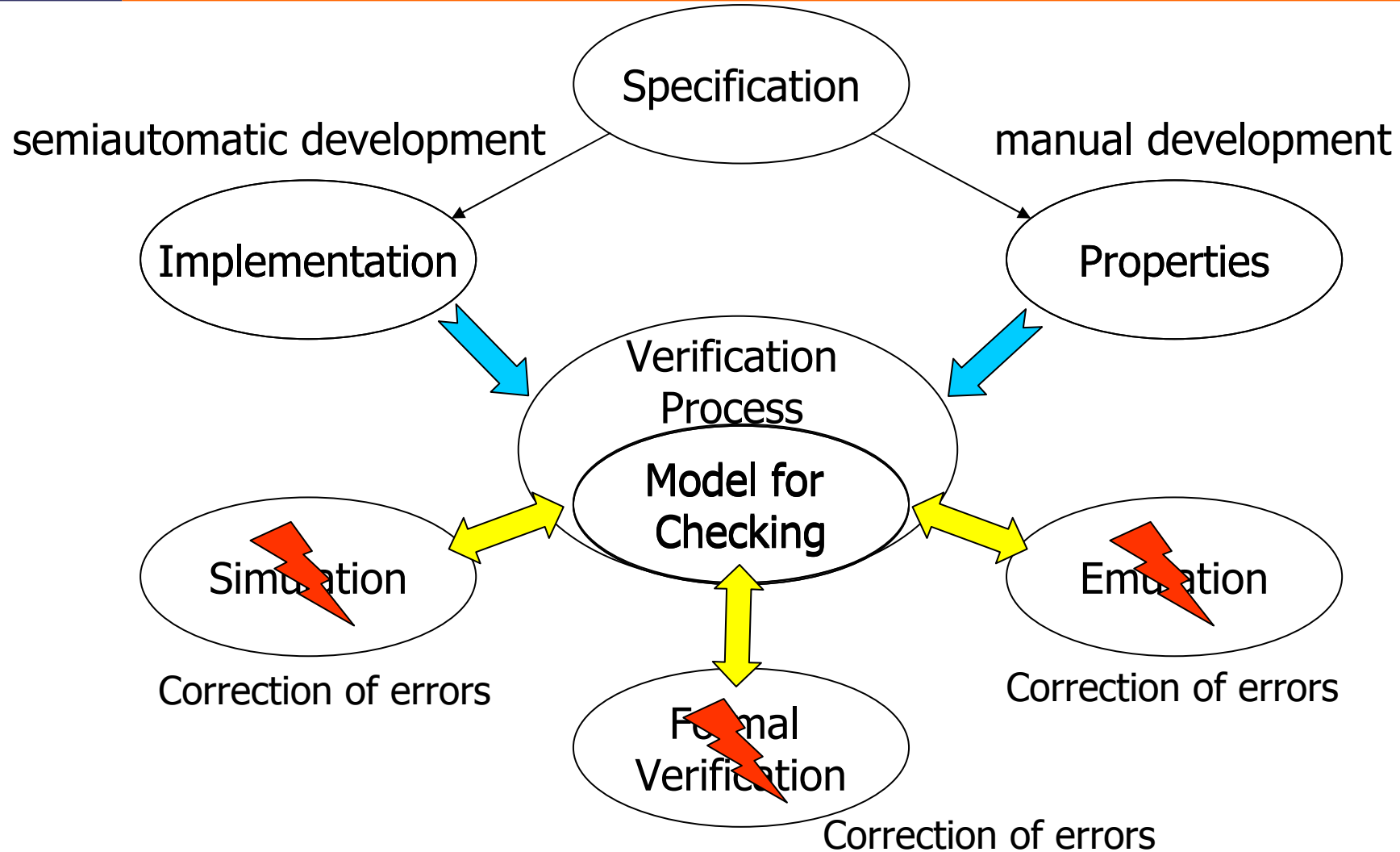
General Verification



Where do the properties for the verification come from?



Verification Process



The whole verification process uses the properties !!!

A *property* is a description of design intent.
[Coelho & Foster, 04]

- Improved understanding of system and its requirements
- Improved communication of design intent among involved parties
- Properties indicate problems close to error source
- The same properties can be used in functional and formal verification
- IEEE Standard 1850: Property Specification Language (PSL)

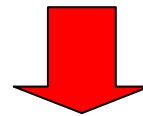


Verification with properties is well suited for system verification.

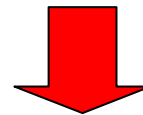


Extraction of Properties

Properties are manually generated for the verification



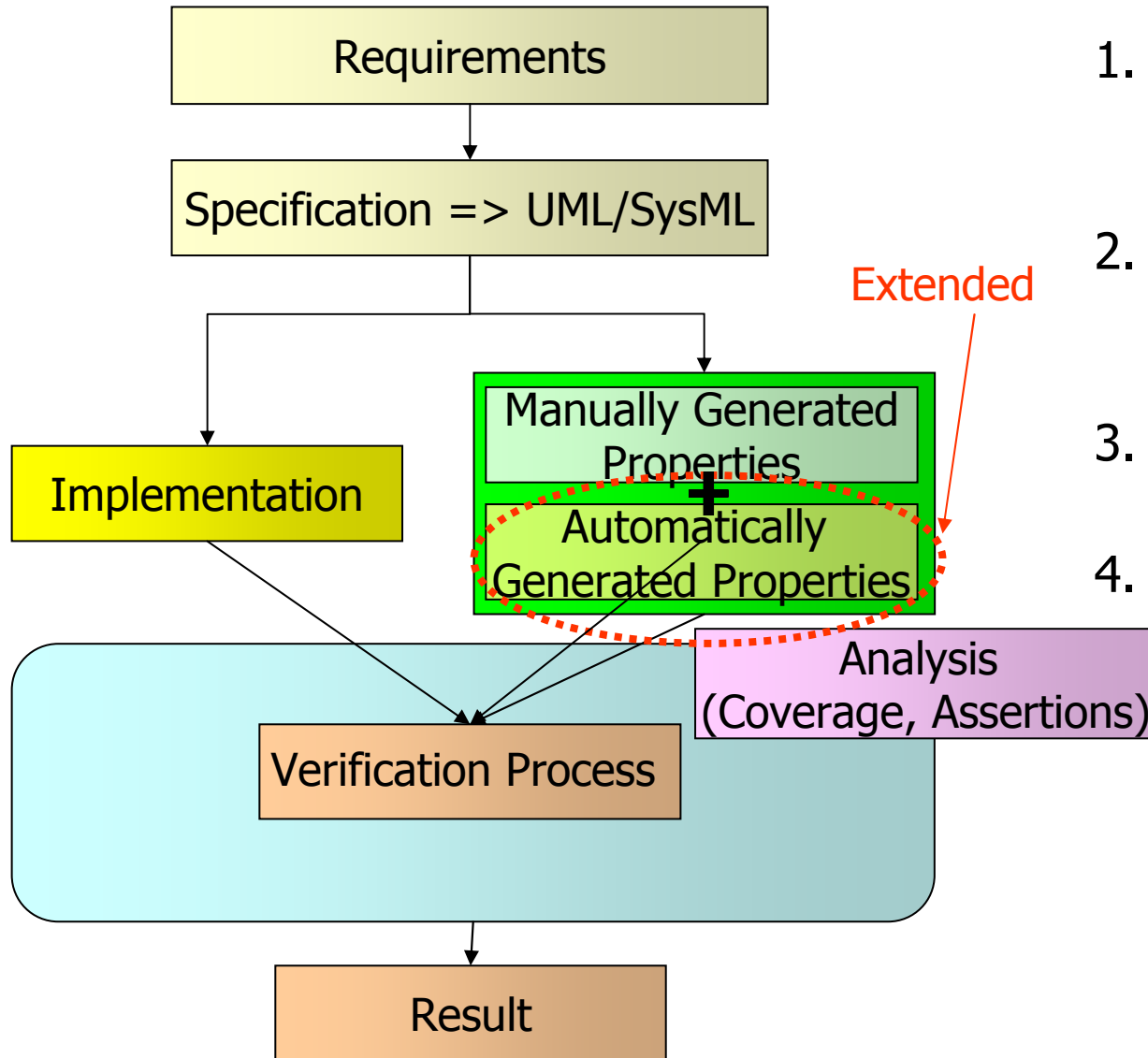
Error-prone – commercial tools have no concept or solution for these problems



→ Automatic extraction of **properties/assertions** support the verification process at different abstraction levels



Idea

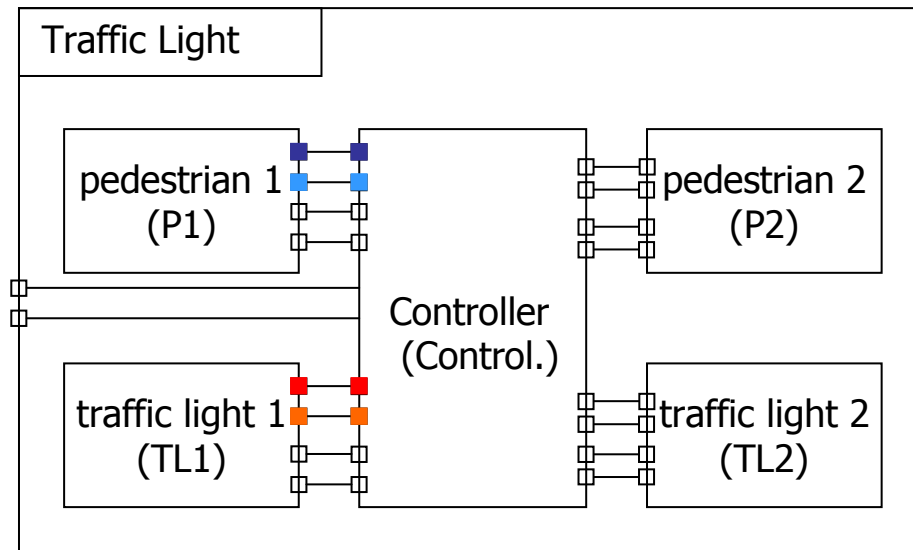


1. UML describes structure and behavior of the system
2. Behavior is the communication between the modules/components
3. Extraction of the properties of the module
4. Application: Simulation and formal verification

PSL Property:

$$\{go_red_TL1; [t_1]; is_red_TL1; [t_2]; go_green_P1; [t_3]; is_green_P1\}$$

Block Definition Diagram



Control. \rightarrow go_red_TL1 \rightarrow TL1
 TL1 \rightarrow is_red_TL1 \rightarrow Control.
 Control. \rightarrow go_green_P1 \rightarrow P1
 P1 \rightarrow is_green_P1 \rightarrow Control.

- Example traffic light
 - ▶ 5 modules (TL1, TL2, P1, P2, Control.)
 - ▶ SysML Design
 - ▶ 1 block definition diagram
 - ▶ 1 sequence diagrams
 - ▶ 6 important properties
 - ▶ 4 directly in the specification
 - ▶ 2 indirectly in the specification (but directly in the design)
- Find manually only 4 properties from the specification (other 2 difficult: implicit in design)
- New method extracts 2 properties, finding all 6 (because it uses the design)
- Method good to locate errors on module level



Applications of the Method



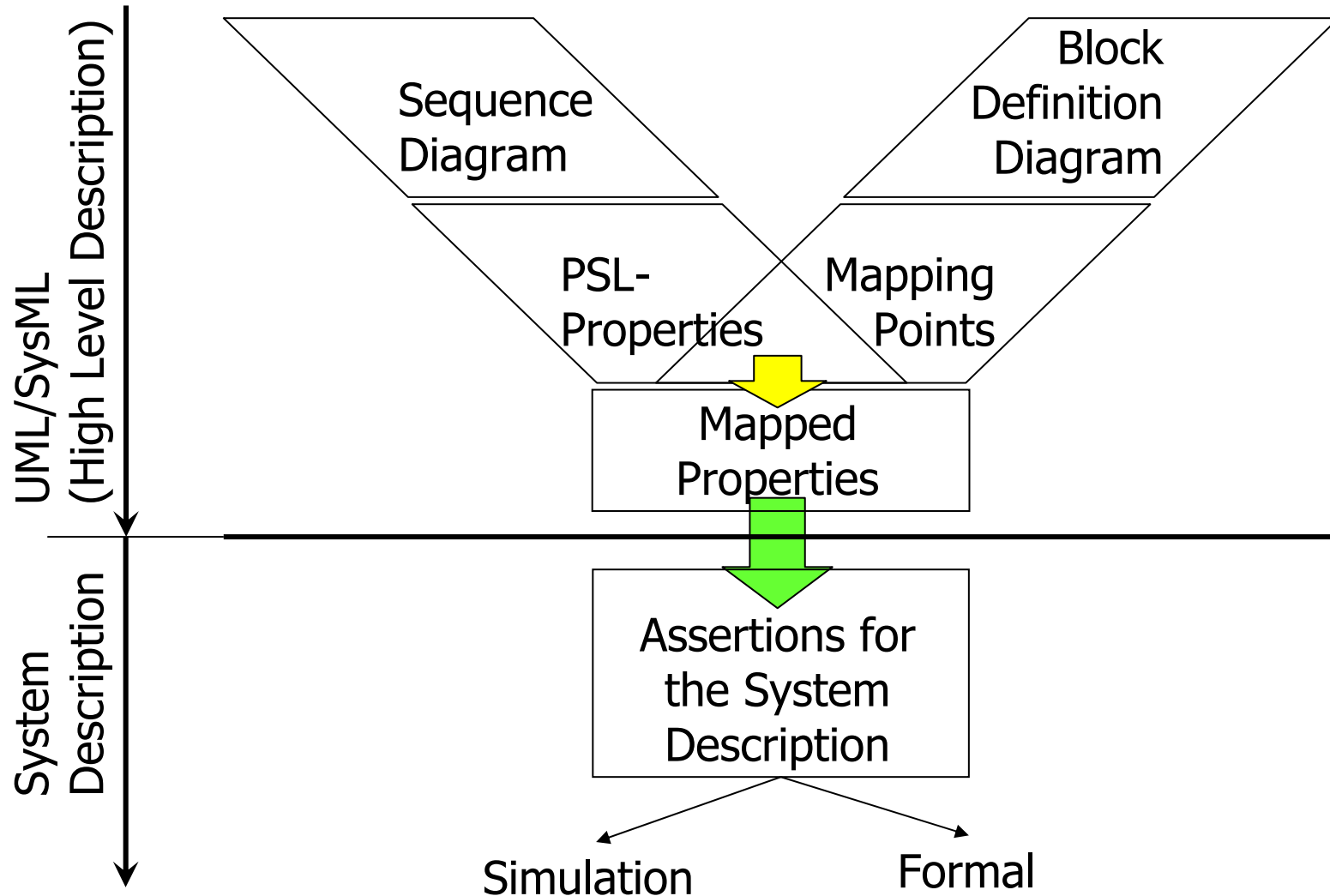
Application Flow

Motivation and Introduction

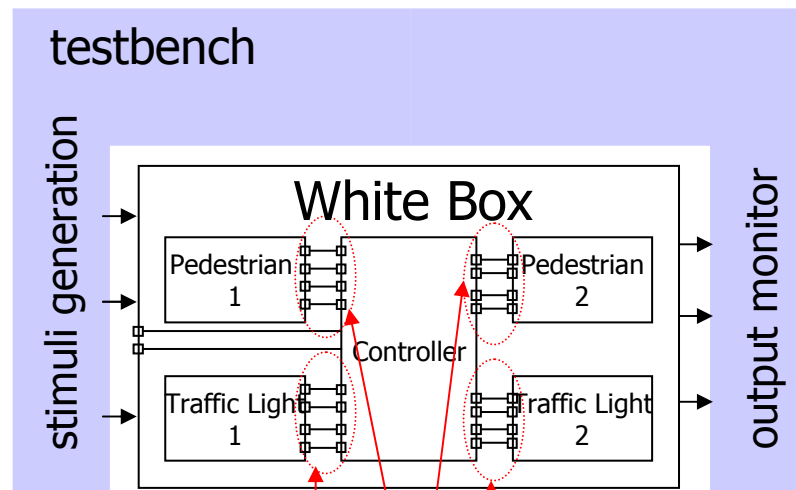
Extraction of Properties

Applications of the Method

Conclusion and Future Work



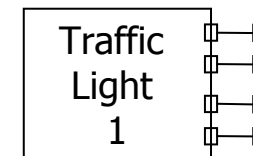
Simulation:



Additionally inserted monitors / assertions watch defined sequences

→ Coverage Analysis

Formal Verification:



$\{go_red_TL1; is_red_TL1\}$

Module Verification:

Formal Verification with a part of the sequence



- Coverage
 - ▶ Complete check of all sequences
 - ▶ Coverage of all sequences
- Efficient generation of simulation sequences
(check for missed sequences)
- Early error detection
- Additional check of time constraints
(in every simulation)



Conclusion and Future Work



Conclusion and Future Work

Motivation and Introduction

Extraction of Properties

Applications of the Method

Conclusion and Future Work

- Problems of System Verification
 - ▶ Finding properties automatically from the specification
 - ▶ Using properties at different abstraction level
- Presented Ideas
 - ▶ Generate properties automatically from system level
 - ▶ Take properties for lower abstraction level
 - ▶ Verify more functional properties
 - ▶ Find additional errors
- Future Work
 - ▶ Assertions for software model-checking
 - ▶ Rating the assertions (important, enough)



Thank you for your attention!

Any questions?