
Towards Verification of Component-based Systems

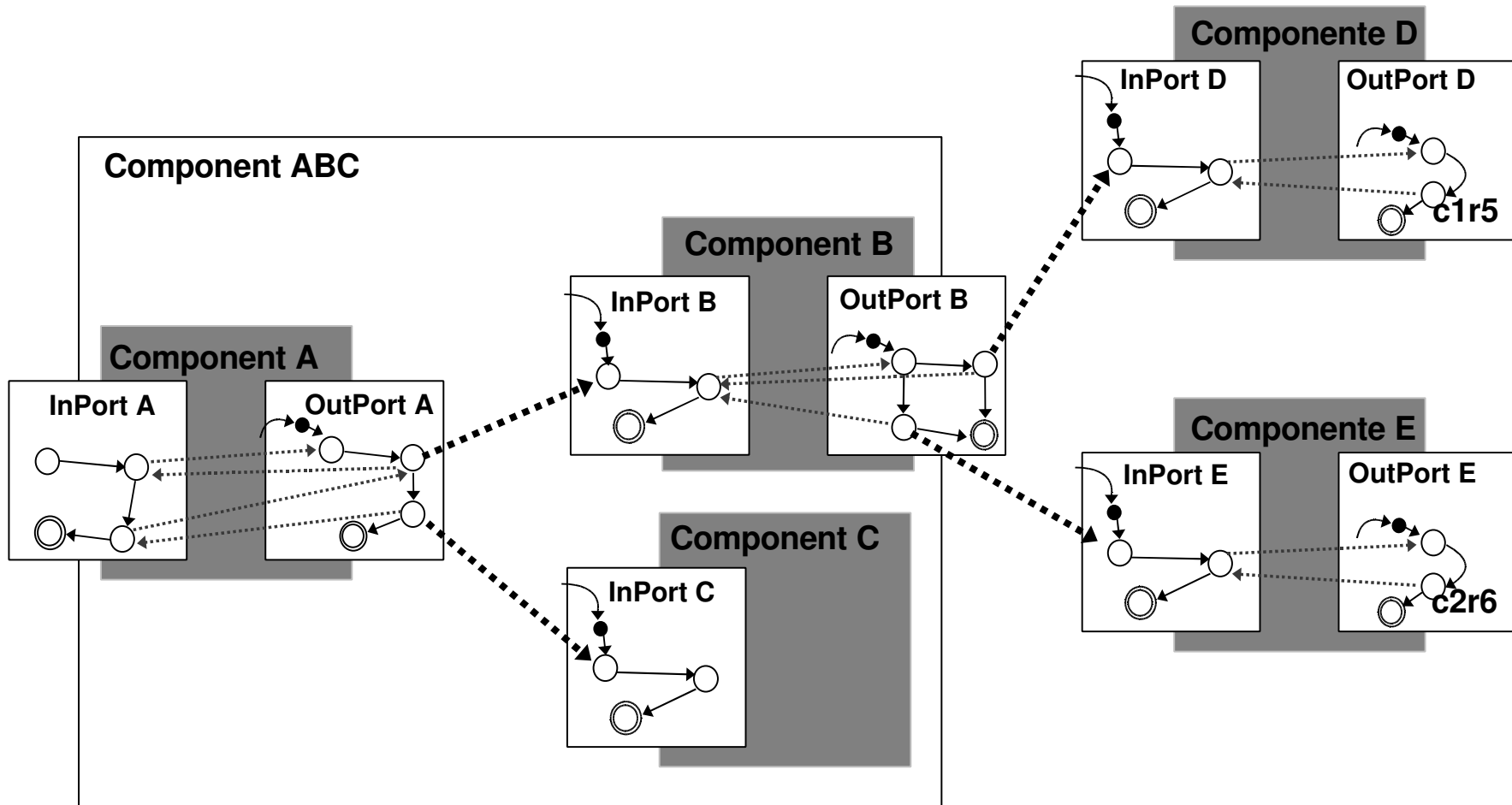
Elke Pulvermüller
Faculté des Sciences, de la Technologie et de la Communication
Université du Luxembourg



Contents

- **Problem Statement**
- **Related Work**
- **Overview of the Approach**
- **User-driven Modeling**
- **Verification-driven Modeling**
- **Summary**

Problem Statement



Problem Statement

Examples for Questions on the System?

Componente D

Dependencies within the static Structure:

Component `<Reviewer>` is connected with the Component `<Submission>` via Messages.



Standard temporal Dependencies:

After the Method `<startReview>` has been initiated in the System calling Method `<submitReviewer>` is not allowed anymore.

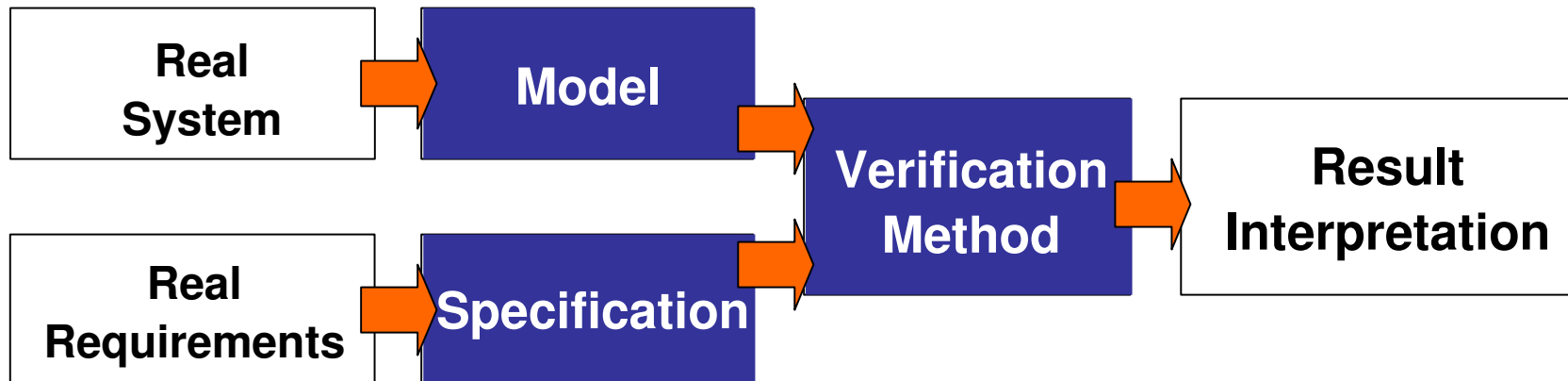


Unknown/Conditions, Annotations, Properties:

Which Conditions have to hold so that the processing time of `<Review>` and `<Reviewer>` is less than 10 unit and this with a probability of > 0.5 ?

Semantic Gaps of the Verification

- **Verification is always „Redundancy Checking“**
“Correctness is the ability of software products to perform their exact tasks, as defined by their **specification.**” [Bertrand Meyer]



Localising the Semantic Gap:

- Modeling Language
- Specification Language
- Verification Algorithms

Problem Statement

Requirements towards the Verification Method

- R1: Mapping of Component System Concepts (e.g. atomic, complex Components, State, Port, Annotation)

- R2: Dealing with „Unknowns“

- R3: Formal Model

Model

Specification

Verification Method

- R6: Automatization

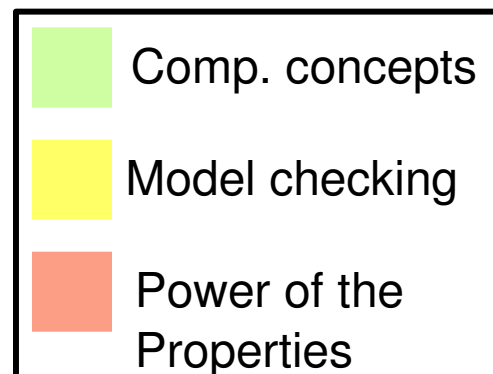
- R7: Extendability (in the Property Types)

- R8: Consideration of „Unknowns“

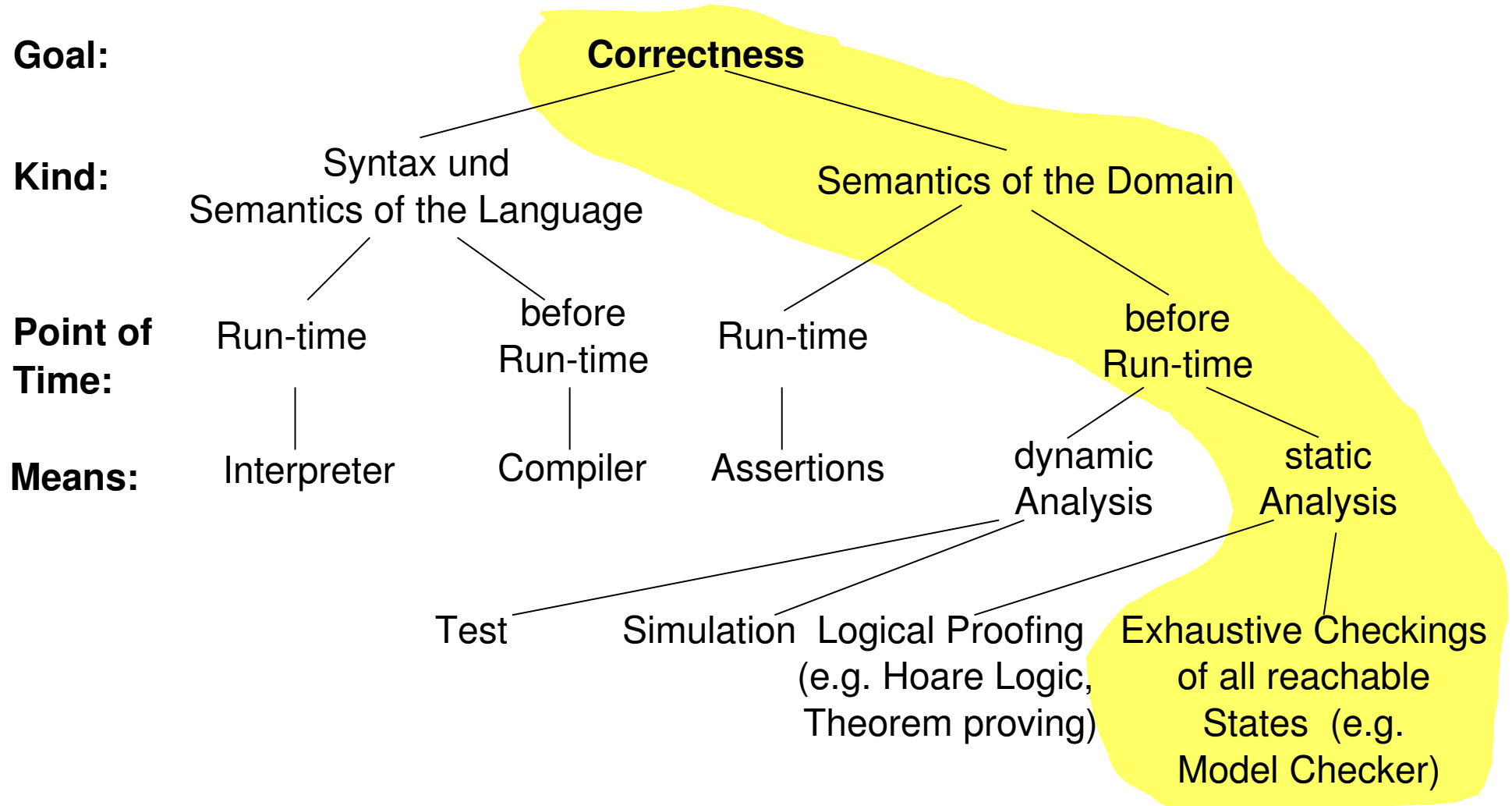
- R9: Scalability

- R4: Temporal Correctness Specification

- R5: Orthogonal, selective Specification over all model elements (also hierarchically)

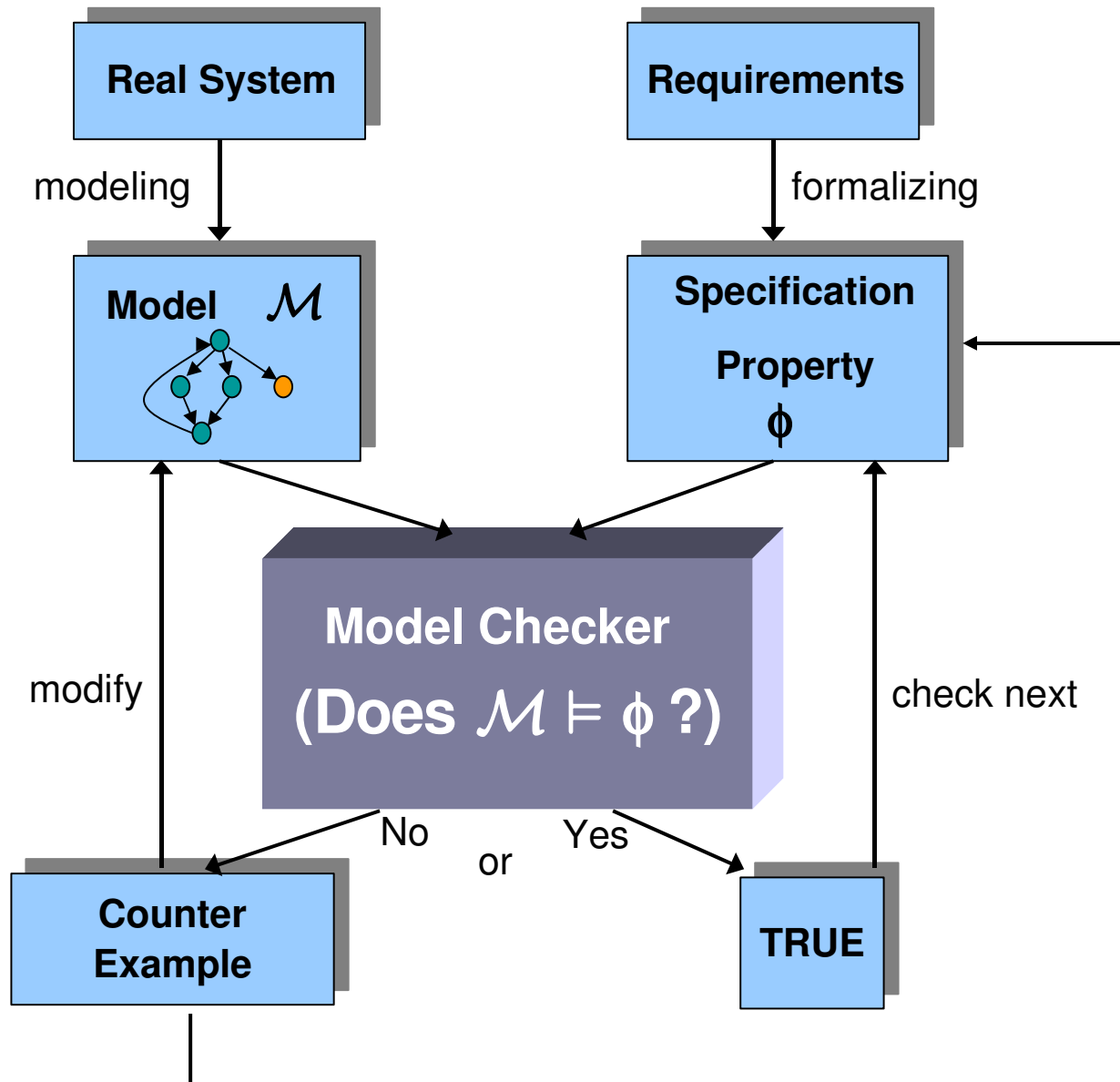


Overview



Related Work

Overview



Related Approaches

Approaches to Modeling and Verification of higher Concepts via Transformation into the input Format of Standard Model Checkers

UML [Eshuis 04], [Lilius 99], ADL [Bose 99], Code [Bandera, JPF], dyn. Allocation [Distefano 01], [Cho 01], infinite State Spaces [Burkart 01]

- ⇒ Specific Concepts
- ⇒ Mapping Problem: injective Mapping leads to Information Loss

Approaches with extended Properties

Assignment [De Nicola 90, Johnsson 90], Labeled Kripke structures [Chaki/Clarke et. al. 04], Time [Alur/Dill 94], UPPAAL [Larsen 97], Hybride [Alur/Henzinger 97]

- ⇒ Lack of Orthogonality: Model Elements are not treated equally w.r.t. Property Assignment
- ⇒ only specific Properties
- ⇒ no specialized temporal Specification Lang.

Approaches with „Unknown“ Properties

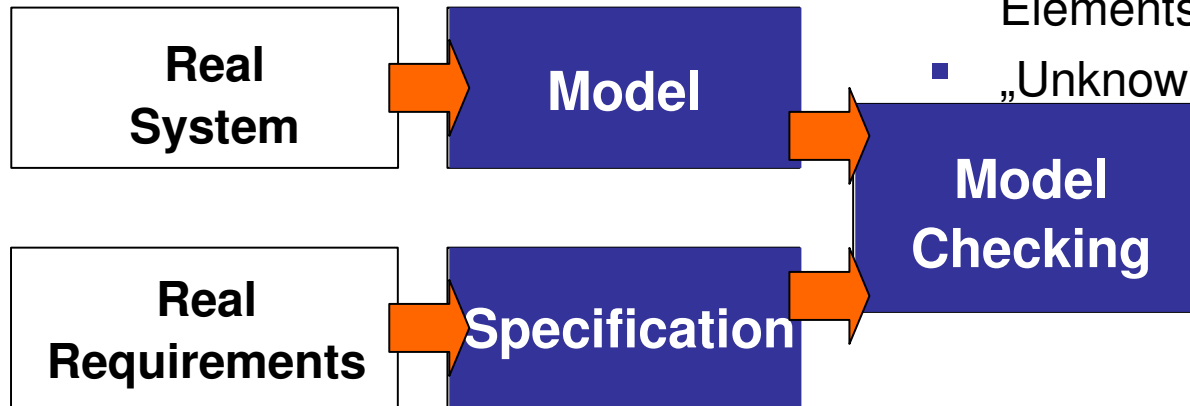
Multi-valued [Huth 01, LNCS 2028], [Bruns/Godefroid 99], [Fisler 04], Probabilities [Hermanns/Katoen 01]

- ⇒ Lack of Scalability

Overview of the Approach

Model close to the Application:

- Frameworks of mathematical Structures



Syntax and Semantic of the Extended temporal Specification Language ECTL (based on CTL):

to define temporal Specifications:

- with Property Hierarchies
- Modelement-awareness (Specializers)
- with „Unknowns“

Extended Automata Model close to the Verification Viewpoint:

Automata Model:

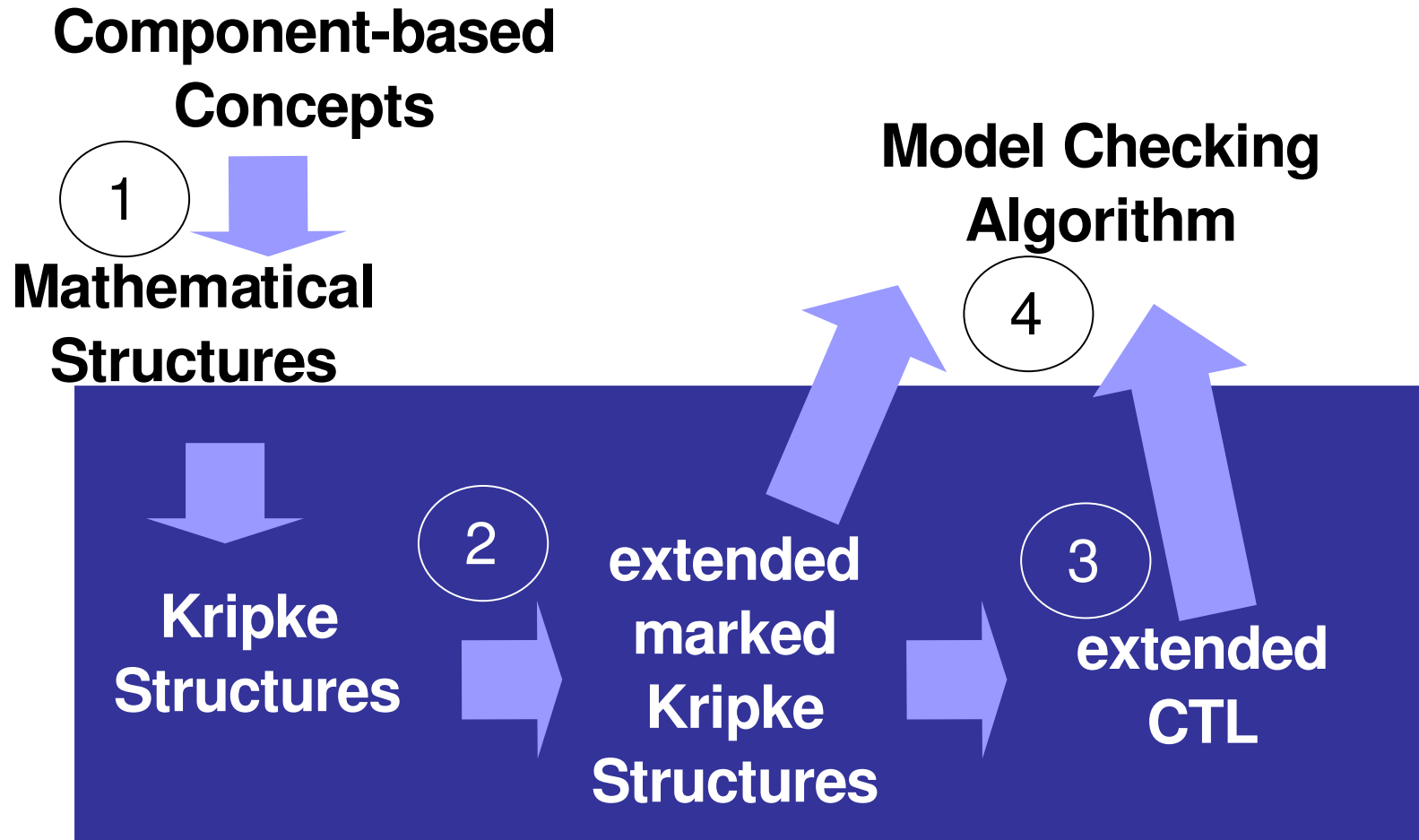
- Property Algebras („Plug-In“)
- Uniformity and Orthogonality of the Mode Elements (Property Assignment)
- „Unknown“

Model Checker CoV

(30 KLOC)

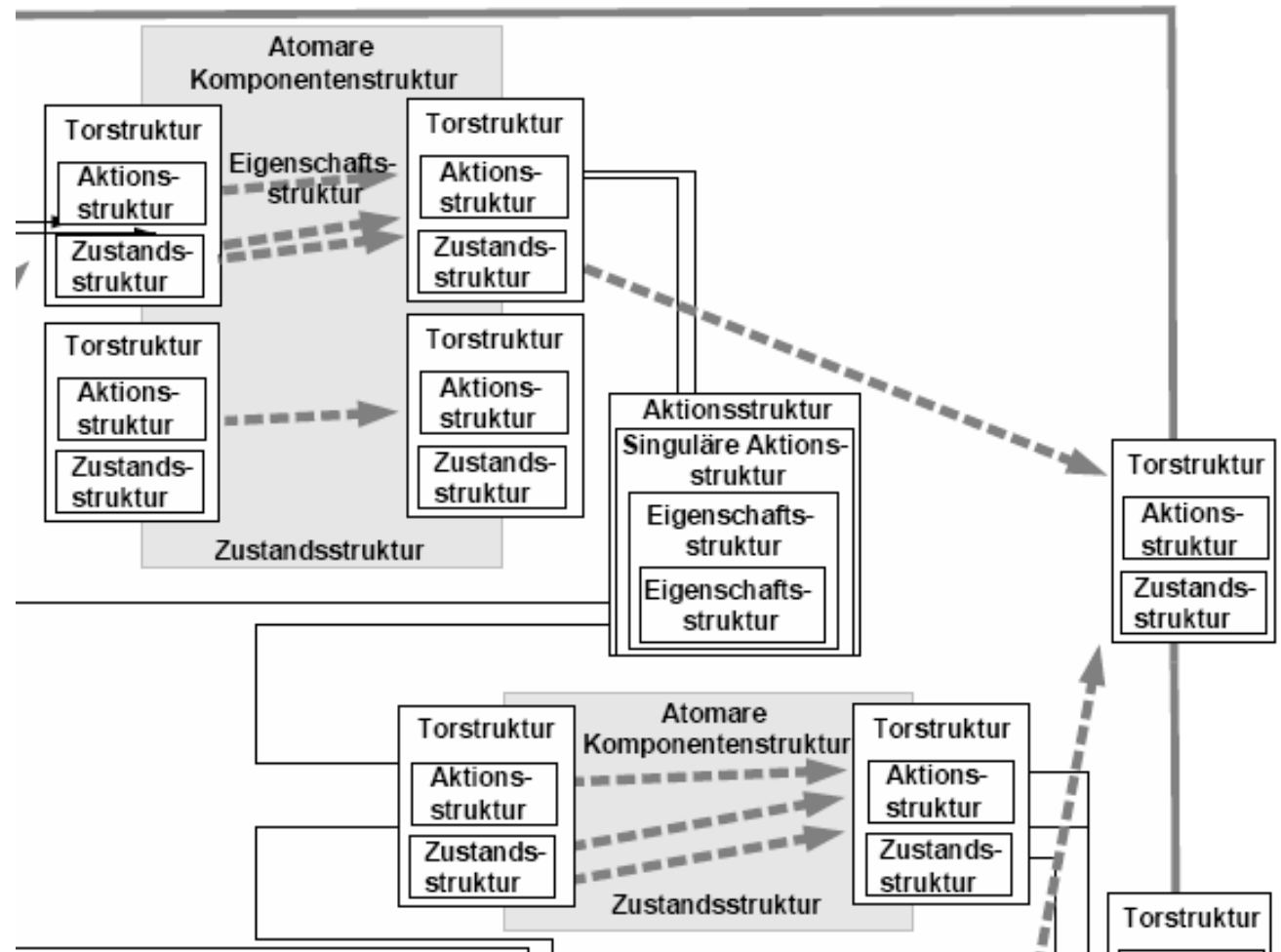
- Adaptation of the Algorithms to Verify ECTL Specifications
- explicit and symbolic (OBDD) Implementation

Overview of the Approach



Frameworks of mathematical Structures (abstract)

- Mapping of Component-based Concepts on mathematical Structures
- Frameworks of mathematical Structures and their Connections (abstract View):



Frameworks of mathematical Structures (abstract)

- Mapping of Component-based Concepts

on
Stru

Port
Structure

Action
Structure

Stru
their

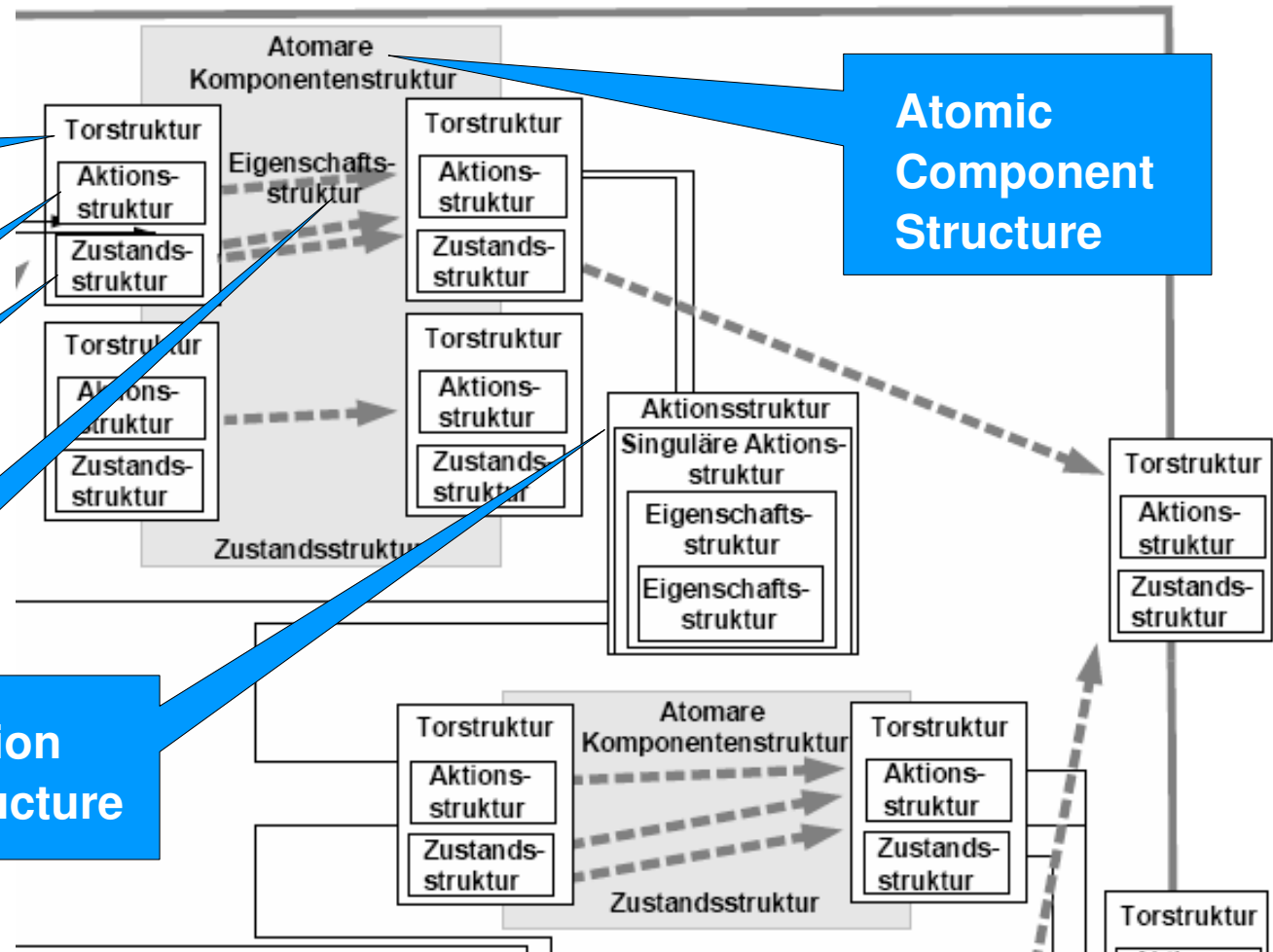
State
Structure

Con
(ab

Property
Structure

Action
Structure

Atomic
Component
Structure



Frameworks of mathematical Structures

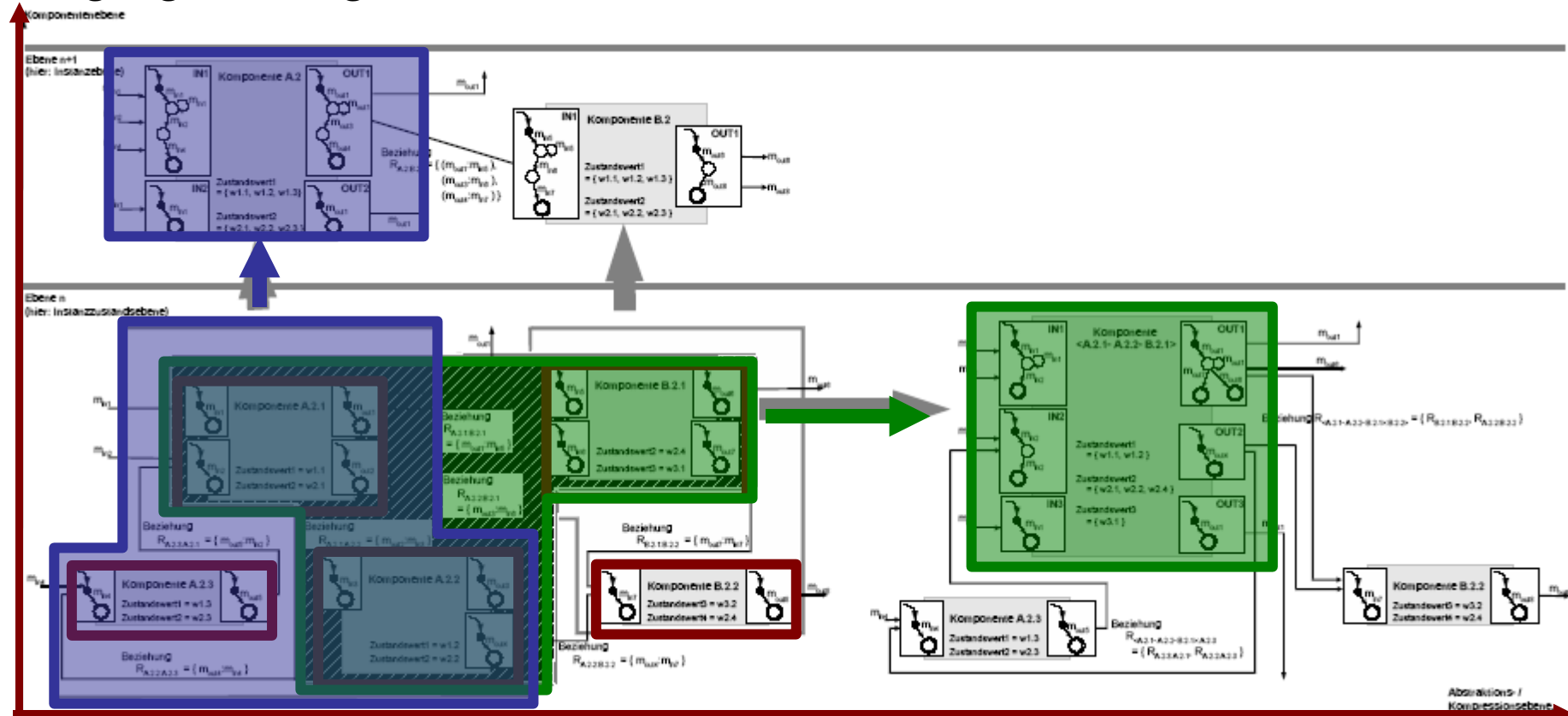
Structure KAC.1 representing the atomic Component C.1:

Sorten: $Z_{KAC.1} = \{ \text{zustand} \mid \text{zustand ist vom Typ Zustandsstruktur } ZC.1 \}$ $T_{KAC.1} = \{ \text{tor} \mid \text{tor ist vom Typ Torstruktur } TC.1 \}$ $E^1_{KAC.1} = \{ \text{eigenschaft} \mid \text{eigenschaft ist vom Typ Eigenschaftsstruktur } EC.1-1 \}$ $E^2_{KAC.1} = \{ \text{eigenschaft} \mid \text{eigenschaft ist vom Typ Eigenschaftsstruktur } EC.1-2 \}$	States, In-/Out-Ports, Properties
Operationen: $\text{verknuepfe}^1_{KAC.1}() = e1$ (mit: $e1$ ist konkrete eigenschaft vom Typ $E^1_{KAC.1}$) $\text{verknuepfe}^{TT^2}_{KAC.1}(za, t1, zb, t2) = e3$ falls ... // für alle anderen gestrichelten Pfeilverbindungen (mit: $e3$ ist konkrete Eigenschaft der Sorte $E^2_{KAC.1}$)	Operations of the Property Algebra Assignment of Properties
Relationen: $\text{einTor} = \{ \text{tor} \mid \text{tor} = \text{IN1.1} \vee \text{tor} = \text{IN1.2} \}$ $\text{ausTor} = \{ \text{tor} \mid \text{tor} = \text{OUT1.1} \vee \text{tor} = \text{OUT1.2} \}$ $\text{paramTor} = \{ \text{tor} \mid \text{tor} = \text{IN1.1} \}$ $\text{ein2aus} = \{ ((za, t), (zb, t2)) \mid za = \text{IN1.1.z2}, zb = \text{OUT1.1.z1}, t1 = \text{IN1.1}, t2 = \text{OUT1.1} \vee \dots // \text{für die anderen gestrichelten Pfeilverbindungen in C.1} \}$	Relations: System Structure (mainly)

Notes:
- Zustandstruktur
- Eigenschaft ist konkrete Eigenschaft mit Prädikat: preis (Beschaffungskosten für die Komponente)
- e2 ist konkrete Eigenschaft vom Typ Eigenschaftsstruktur e3 hat Prädikat: TRUE

Abstraction Alternatives

Abstraction along the Language Paradigm

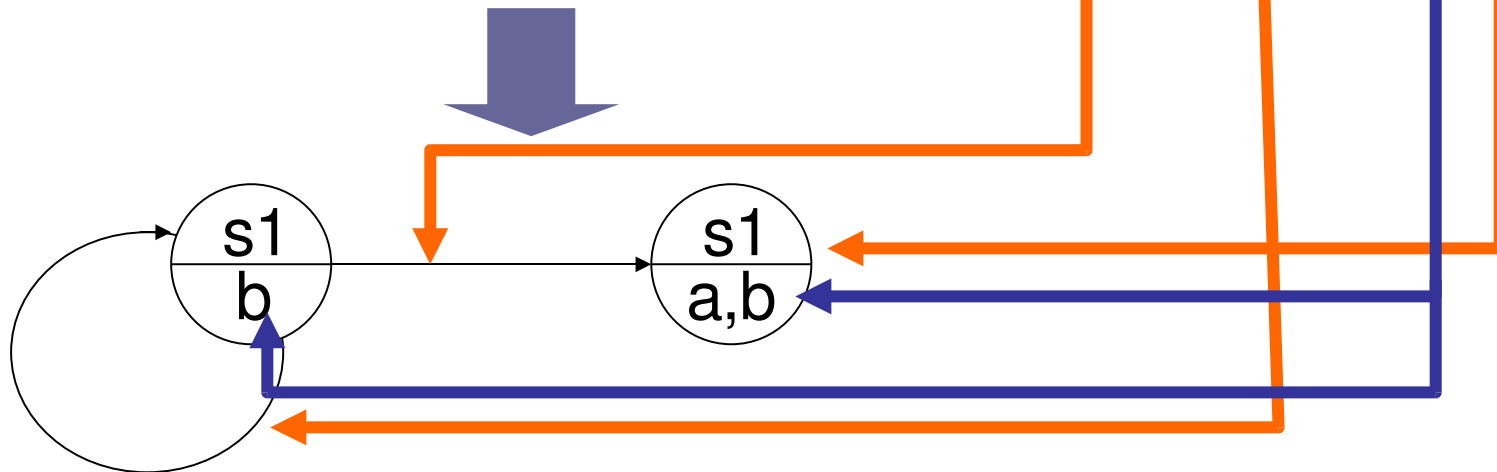


Abstraction within the Paradigm

Verification-driven Modeling

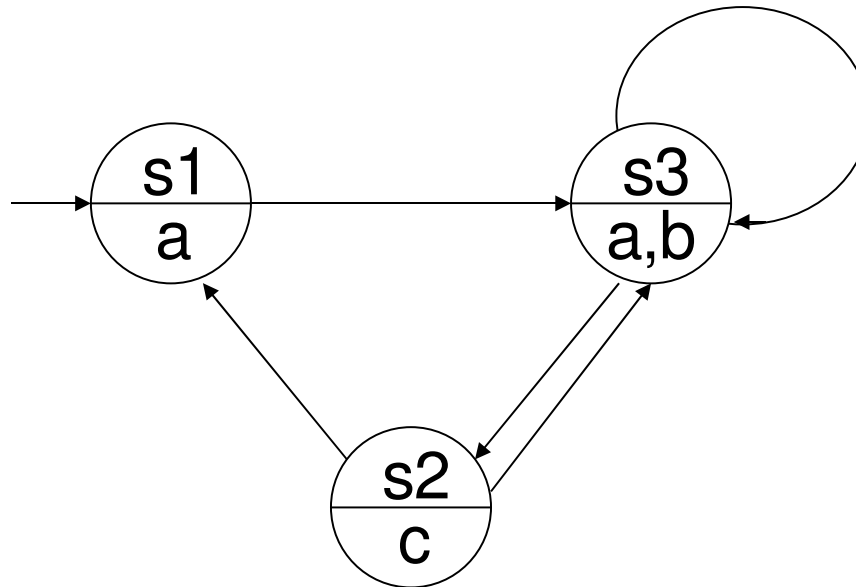
Mapping Principle: from mathematical to Kripke Structures

Sorts:	$\text{sort}_1, \text{sort}_2$	$\text{sort}_1 = \{s1, s2\},$ $\text{sort}_2 = \{a, b, c\}$
Operations:	$\text{label}: \text{sort}_1 \rightarrow \text{sort}_2$ $\text{add}: \text{sort}_2 \times \text{sort}_2 \rightarrow \text{sort}_2$	$\text{label}(n) = p$ $\text{add}(n_1, n_2) = n_1 + n_2$
Relations:	$r_1: \langle \text{sort}_1, \text{sort}_1 \rangle$	$r_1 = \{ (s1, s2), (s1, s1) \}$



Kripke Structure Extensions needed - Base

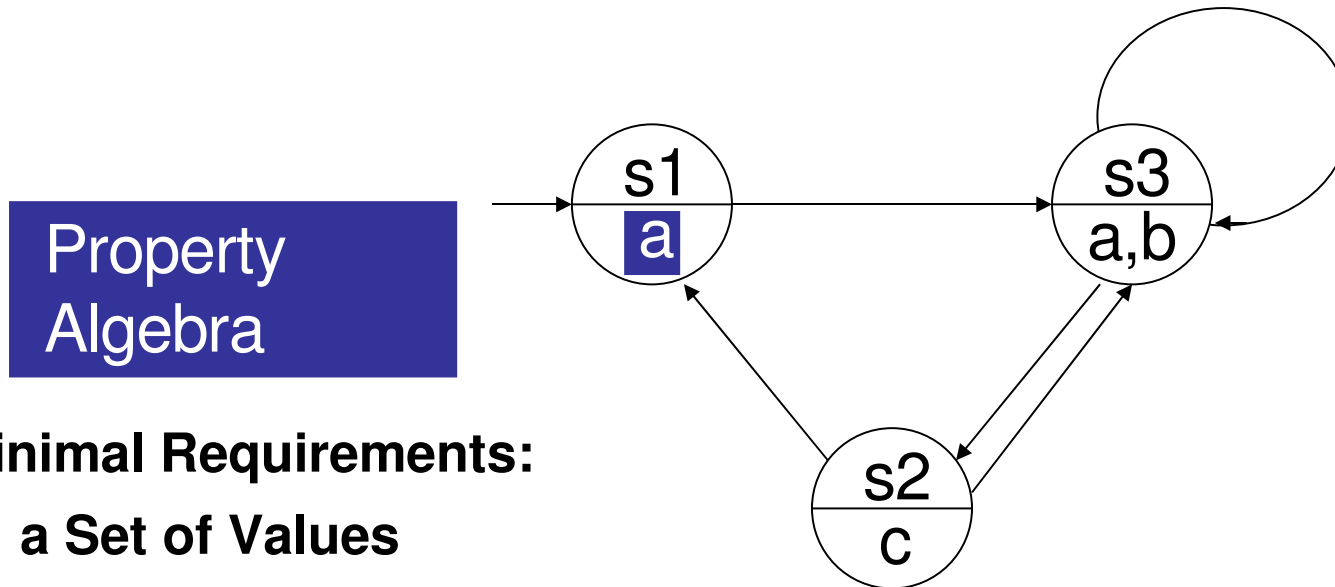
Kripke Structure \longrightarrow Kripke Structure, extended



$$K=(S,S_0,R,L_S,AP)$$

Kripke Structure Extensions needed - Property Algebra

Kripke Structure \longrightarrow Kripke Structure, extended



Minimal Requirements:

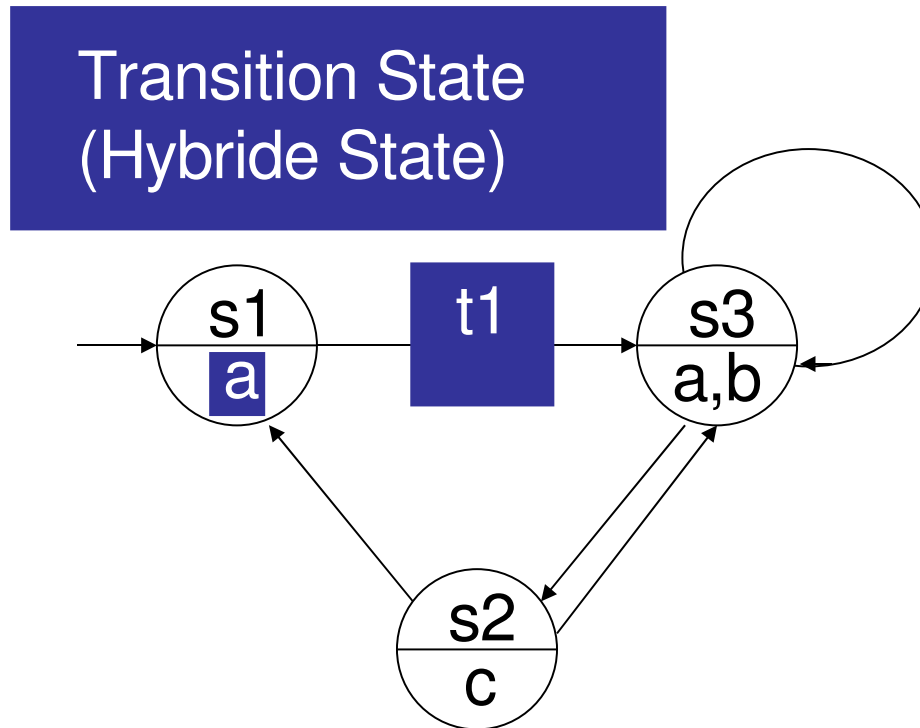
- a Set of Values
- Three basic Operations
AND, OR, COMPARE
- Quasi-Boolean Algebra

$$K = (S, S_0, R, L_S, P)$$

$$P = P_1 \cup \dots \cup P_n$$

Kripke Structure Extensions needed – Transition State

Kripke Structure \longrightarrow Kripke Structure, extended



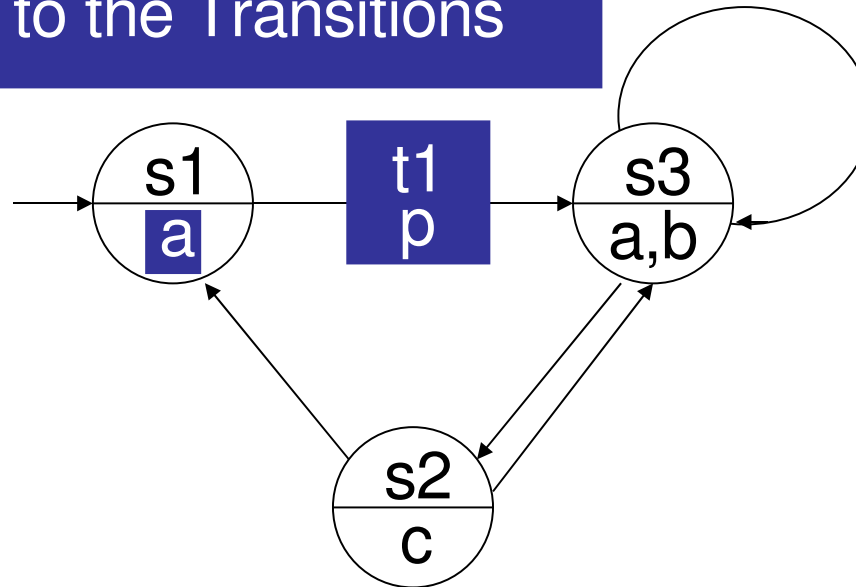
$$K = (S, T, S_0, L_S, P)$$

$$P = P_1 \cup \dots \cup P_n$$

Kripke Structure Extensions needed – Property Assignment

Kripke Structure \longrightarrow Kripke Structure, extended

Properties assigned
to the Transitions

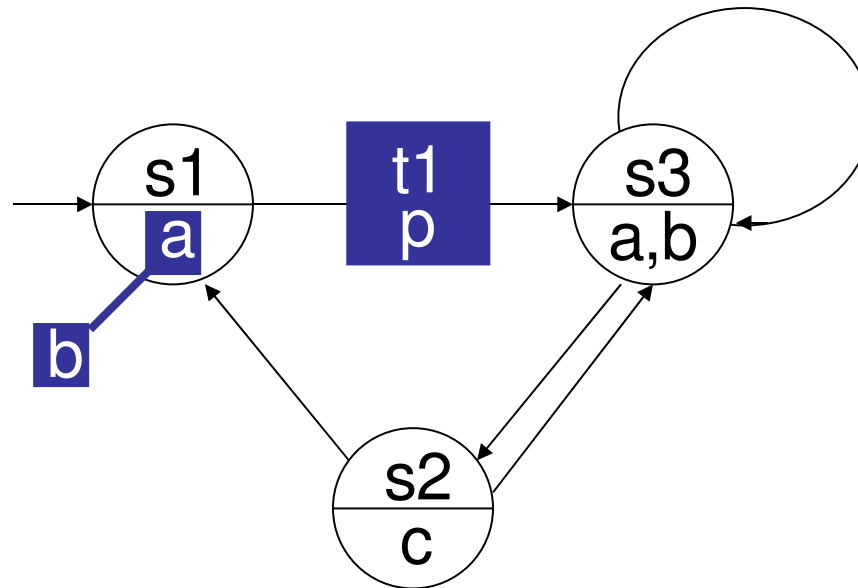


$$K = (S, T, S_0, L_S, L_T, P)$$

$$P = P_1 \cup \dots \cup P_n$$

Kripke Structure Extensions needed – Property Assignment

Kripke Structure \longrightarrow Kripke Structure, extended



Hierarchical
Properties

$$K = (S, T, S_0, L_S, L_T, L_P, P)$$

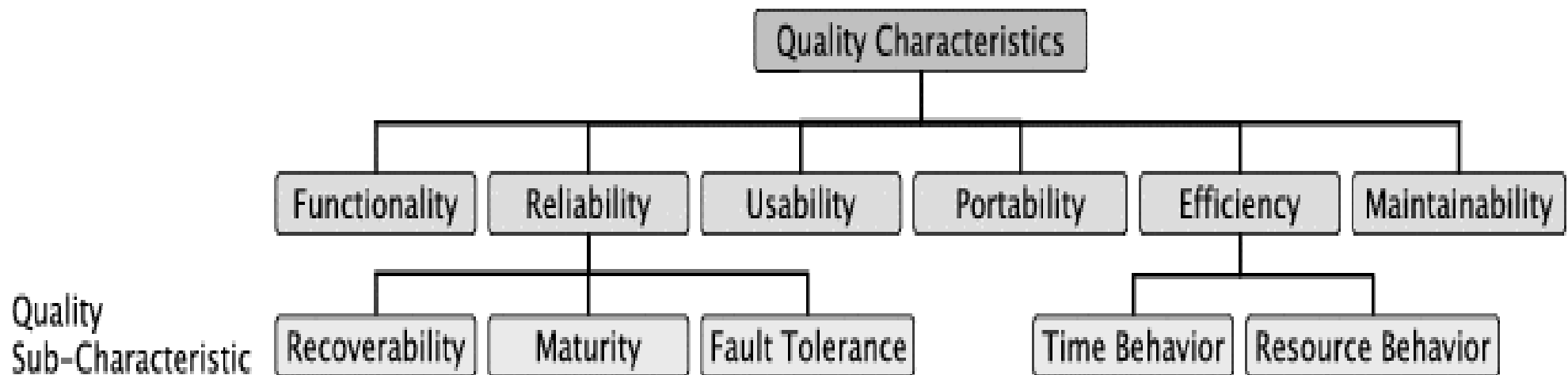
$$P = P_1 \cup \dots \cup P_n \cup P_p$$

Kripke Structure Extensions needed – Property Assignment

Application Example for Hierarchical Properties: [REMO2V WS 06]

- Quality Properties as Rules / Regulations augmenting the Model
- Regulations often refer to multifarious and structured properties
- Insertion of additional non-functional Properties into BPEL4WS Model Definition

Excerpt of the ISO 9126-1 Quality Model



Verification-driven Modeling

Kripke Structure Extensions needed – „Unknowns“

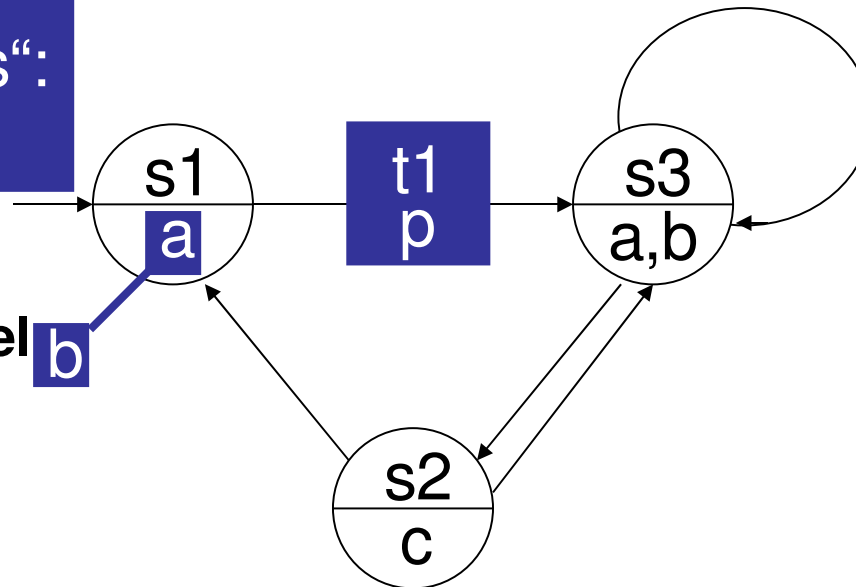
Kripke Structure \longrightarrow Kripke Structure, extended

Special Property Type for „Unknowns“: Conditions

- determines the Shape of the Model
- requires special Treatment in the Model Checker

Conditional Satisfaction Function

$[C]M, s \models \Phi \text{ iff } C = \text{TRUE}$



$$K = (S, T, S_0, L_S, L_T, L_P, P)$$

$$P = P_1 \cup \dots \cup P_n \cup P_p \cup P_u$$

CTL Language Extension: ECTL

CTL Extension (Syntax + Semantics): ECTL

$$\Phi ::= B \mid \text{HSP} \mid \text{SP} \mid \text{BC} \mid \text{TC}$$

with

Base Value: $B ::= \perp \mid \top \mid p$

Hierarchically specialized Properties: $\text{HSP} ::= p \mid P[\text{HSP}, B]$

State Specializers: $\text{SP} ::= S[\Phi] \mid T[\Phi]$

Boolean Connectors:

$$\text{BC} ::= (\neg\Phi) \mid (\Phi \wedge \Psi) \mid (\Phi \vee \Psi) \mid (\Phi \rightarrow \Psi)$$

Temporale Connectors:

$$\begin{aligned} \text{TC} ::= & \text{AX } \Phi \mid \text{EX } \Phi \mid \text{A}[\Phi \text{ U } \Psi] \mid \text{E}[\Phi \text{ U } \Psi] \\ & \mid \text{AG } \Phi \mid \text{EG } \Phi \mid \text{AF } \Phi \mid \text{EF } \Phi \end{aligned}$$

Model Checking Adaptations

- dealing with the new ECTL semantics and the specializer
- using the (plug-in) property algebras and hierarchical properties in the model
- conditional satisfaction relation $[C]M, s \models \Phi$ iff $C = \text{TRUE}$

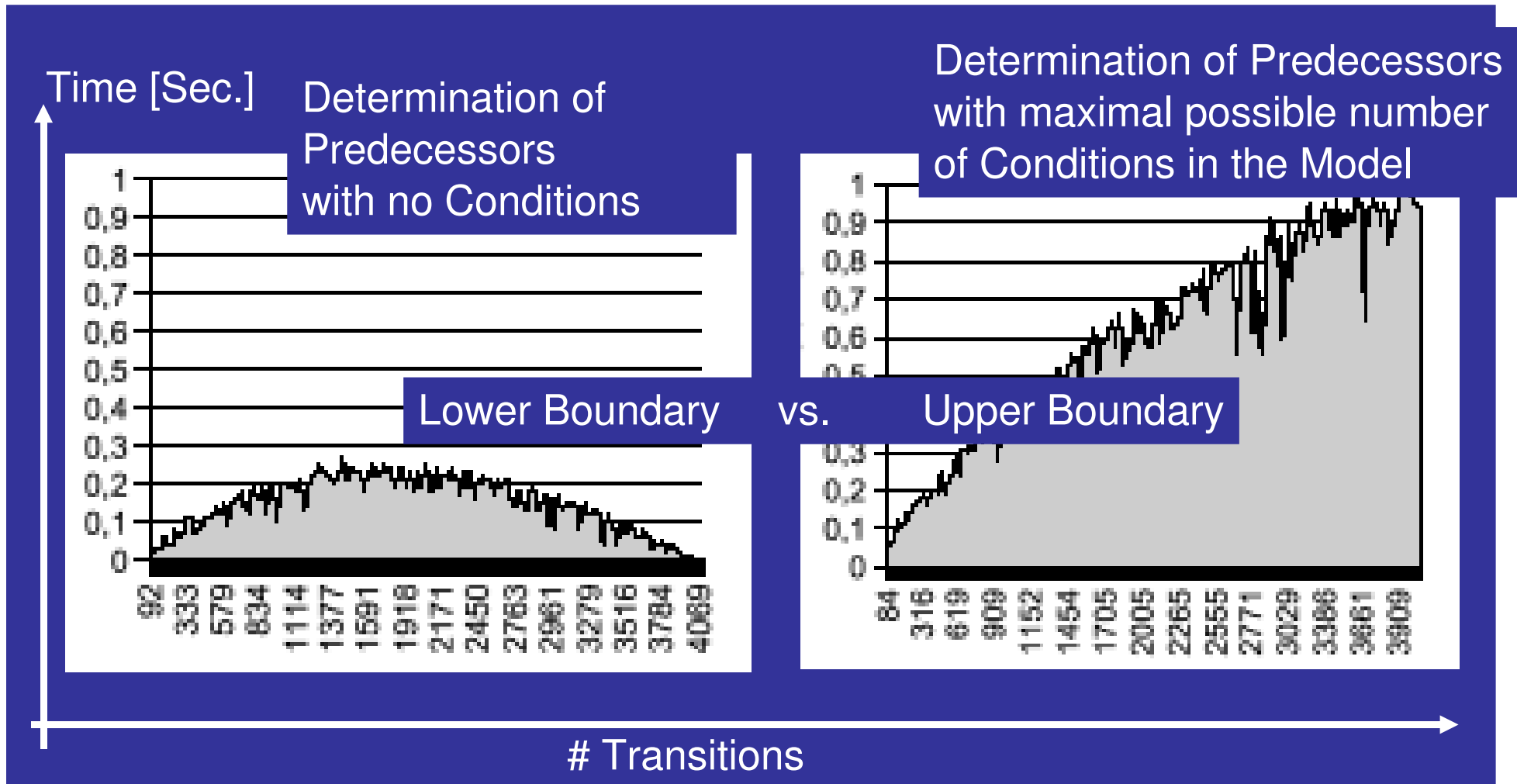
result depends on the logic condition expression $[C]$

example result (CoV implementation):

$\{ s1[c1 \wedge c2], t1[c4], s4[c1 \vee (c3 \wedge c4)], s7[\text{TRUE}] \}$

Verification-driven Modeling

Impact of Conditions on symbolic Verification



Verification-driven Modeling

Example Application of CoV

Output while processing input.txt

Definition of the Model and Specification: input.txt

```
-- Definition des System-  
-- modells:  
MODULE main  
VAR  
  m0 : boolean;  
  m1 : boolean;  
  m2 : boolean;  
  m3 : boolean;  
INIT  
  m0 := x1;  
  m1 := x2;  
  m2 := ;  
  
ASSIGN  
m0 -t-> m1 conditions: c1;  
m1 -t-> m2 conditions: c2;  
m2 -t-> m2 conditions: c3;  
m2 -t-> m0 conditions: c4;  
MODEND  
  
-- Spezifikation der  
-- temporalen Korrektheits-  
-- anforderung:  
SPEC  
AF x2;
```

```
/home/user/bin> cov -uCond -uo -o output.txt ../examples/input.txt  
  
Result file is set to: output.txt  
Opening input file: ../examples/input.txt  
Success: input file ../examples/input.txt has been opened  
cov is model-checking ../examples/input.txt ...  
[main] Created new symbol table: 8082908  
[main] Created new parse tree: 8082950  
[main] Created new spec parse tree: 8082998  
  
[main] Initialising the specification and connected data structures ...  
[main] Initialising the model set and connected data structures ...  
  
[main] Start model checking ...
```

<snip>

```
[StateSet] State set = { m1[x2], m0[x1], m2[] }  
[StateSet] Conditions for state: m1[x2]  
[ConditionSetSet] ConditionSet set = { }  
  
[StateSet] Conditions for state: m0[x1]  
[ConditionSetSet] ConditionSet set = { [c1] }  
  
[StateSet] Conditions for state: m2[]  
[ConditionSetSet] ConditionSet set = { [c1 AND c4 AND !c3] }
```

Verification-driven Modeling

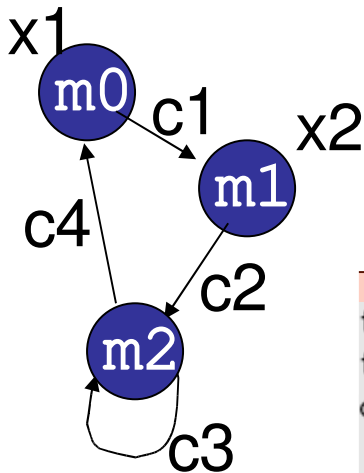
Example Application of CoV

Beispiel (CoV)

Definition of the Model and Specification: input.txt

```
-- Defin
-- model
MODULE m
VAR
  m0 : b
  m1 : b
  m2 : b
  m3 : b
INIT
  m0 :=
  m1 :=
  m2 :=
ASSIGN
m0 -t->
m1 -t-> m2 conditions: c2;
m2 -t-> m2 conditions: c3;
m2 -t-> m0 conditions: c4;
MODEND

-- Spezifikation der
-- temporalen Korrektheits-
-- anforderung:
SPEC
AF x2;
```



Output while processing input.txt

```
/home/user/bin> cov -uCond -uo -o output.txt ../examples/input.txt

Result file is set to: output.txt
Opening input file: ../examples/input.txt
Success: input file ../examples/input.txt has been opened
cov is model-checking ../examples/input.txt ...
[main] Created new symbol table: 8082908
[main] Created new parse tree: 8082950
[main] Created new spec parse tree: 8082998

[main] Initialising the specification and connected data structures ...
[main] Initialising the model set and connected data structures ...

[main] Start model checking ...
```

<snip>

```
StateSet] State set = { m1[x2], m0[x1], m2[] }
StateSet] Conditions for state: m1[x2]
ConditionSetSet] ConditionSet set = { }

[StateSet] Conditions for state: m0[x1]
[ConditionSetSet] ConditionSet set = { [c1] }

[StateSet] Conditions for state: m2[]
[ConditionSetSet] ConditionSet set = { [c1 AND c4 AND !c3] }
```

Problem Statement:

Semantic Gap between Real Systems and formal Verification

⇒ Modeling / Formalization with mathematical Structures and more powerful Property Treatment

Verification Method Elements:

- Formalization of the Concepts of the Component-based Systems (Frameworks of mathematical Structures)
- Extended Kripke-Structures
- Extended Temporal Logic Language CTL (ECTL)
- Extended Model Checking Algorithms / conditional Model Checking

Future Work Areas:

- Support to guide the Abstraction (e.g. based on Metrics)
- Compositional Verifications
- Application to System Generation (Integrity rules)
- Integration into well-known Model Checker (e.g. SMV)
- Optimization (Run-Time, Memory, partial Evaluation of Conditions)
- ...