Introduction to Formal Methods

Ruurd Kuiper

June 26, 2006

Contents

Chapter 1

Introduction

In this course we address formal methods in software development. We have a dual aim: providing an introduction to several methods and providing an assessment of such methods.

We use D. Peled, Software reliability methods, Springer 2001, ISBN 0-387-95106-7. We refer to the book by DP. ... .

To make these aims more concrete, we take as a running example a small system and some of its properties. In this first lecture, we introduce the example informally, formalizing it in various ways later on.

From this example we arrive at a preliminary table of development formalization issues and a model for the stages in the process of developing such a system.

Finally, we present a tentative proposal for the upcoming lectures.

1.1 An algorithm

We discuss formalization aspects of a mutual exclusion algorithm for two processes with a critical section. We first represent it informally as a picture with nodes and arrows: the nodes intuitively are states, the arrows are transitions between the states, with guards on them to indicate when a transition is enabled.

The desired properties we also present informally.

1. Mutual exclusiveness (ME): processes are not simultaneously in their critical section.
2. Eventual access (EA): if a process tries to enter its critical section it eventually will get access.

Various attempts are shown:

1. Attempt S1: only guards on the transition to the critical section, forbidding access if the other is in the critical section.

	syntax	semantics	abstraction	compositionality	automation
System	?	?	?	?	?
Connection	?	?	?	?	?
Specification	?	?	?	?	?

2. Attempt S2: first come first serve variables.
3. Attempt S3: arbitrary but finite stay in critical section.
4. Attempt EWD: DP.83

pictures given at lecture, to be added to notes

1.2 A preliminary table

From the attempts we derive some ingredients for a development formalism.

1. Formalization of the system - syntax and semantics.
2. Formalization of the (properties) specification - syntax and semantics.
3. Formalization of the connection between system and (properties) specification - syntactic and semantic.

Furthermore, to deal with the complexity of the development task, we observe some desirable features of a development formalism.

1. Abstraction.
2. Compositionality.
3. Automation

This leads to a preliminary table for assesment of development formalisms (we place the connection between gthe two connected things - the System and the Specification):

Exercise: Put the various Attempts in the table.

1.3 A model for the process

The table for development formalisms list the ingredients of formalisms. It does not address which formalism should be applied for which task, in fact, no tasks have been identified. We follow a well-established approach to identify and order development tasks, with the following ingredients.

1. Artifacts: tangible items that are produced during the development. For example, the descriptions of system properties, the designs, the implementations, etc.
2. Activities: the activities to obtain the artifacts. For example, to obtain the implementation of a component from a specification, to design a test, etc.

These artifacts and activities are usually ordered in some way: as a waterfall model, as a V-model, as spiral model, as a Rational Unified Process adaptable model ... We follow the approach where the artifacts are ordered as a V (the activities are assumed to take place between the various artifacts, to produce them).

The idea is, that the ordering indicates information flow.

In its most simple form [P.E. Brook 1986], the V's left branch, going down, covers

- Requirement specification
- Structural design
- Detailed Design
- Implementation

with corresponding entries on the right branch, going up,

- Unit test
- Integration test
- Acceptance test
- Operation and maintenance

Note, that apart from information flow between adjacent boxes in the V, also across opposite boixes in the right and left branches in formation flows, as the tests are derived from/c arried out against the artifacts in these boxes.

Exercise: Put the various Attempts in the V's.

1.4 Tentative proposal for further lectures

We address the following topics.

1. Modelling the system: Transition system.
2. Specification and verification: Hoare Logic

3. Specification: Linear time temporal logic (LTL). Specification: Branching time temporal logic (CTL, CTL*) (In model check chapter)
4. Verification automated - semantically: Computation tree logic (CTL) model checking.
5. Specification automated - syntactically: Hoare logic and (i) PVS (ii) ESC/Java
6. Testing
7. Formalism with different abstraction: Process algebra
8. Specification with different compositionality Dijkstra/Hoare logic for object orientation (automated).
9. Application of formal methods and automation at imTech.

Chapter 2

System model: Transition system

We present a model for systems: Transition systems

DP.4: Introductory remarks chapter 4, 4.1, 4.2, 4.3, 4.4, 4.6 Mutual Exclusion4.8, 4.10, 4.11, 4.12

2.1 Specification

DP.71 4.4 is the crucial section: Syntax formalized as transition systems, semantics formalized as state sequences.

Check whether you are familiar with the contents of DP chapter 2 and 3 as far as needed for understanding the formal definition of transition systems.

2.2 Mutex as a transition system

DP.83 This is the crucial example.

Note, that transition systems, i.e., with guarded transitions, are used as syntax. This is because we want an intuitively understandable model for systems: in such systems guards occur, e.g., in while statements.

The semantics is state sequences, without explicit guards being present on the state transitions.

Often transition system without guards can be used to give a finite representation of these sequences, but not always, e.g., when the statespace is infinite. Exercise DP.4.6.1 suggests that a transition system without guards would also give a finite description for the mutual exclusion algorithm.

	syntax	semantics	abstraction	compositionality	automation
System	trans. system	set state seq's	no system structure	global	?
Connection	-	-	-	-	-
Specification	-	-	-	-	-

2.3 Assessment

2.3.1 The matrix

Transition systems are (a little differen from the position taken at the lectures) viewed as a syntax for systems, with sets of state sequences as semantics. They abstract away from the structure of the system, for example in Mutex the parallellism is not represented - therefore the modelling is global as well. Automation is optional: it is possible, e.g. for prototyping purposes, to implement transition systems.

2.3.2 The V's

Transition systems fits in the Realization-V as an artifact at the level of Specification and Design where it acts as a model for the system. If the transition system model is implemented as an executable model, it can be used as an artifact in the V&V-V at the same level.

Chapter 3

Specification and verification: Hoare logic

A syntax and semantics for property specifications for systems and properties can be found in DP 7, notably DP.7.4: We assume familiarity with Hoare logic as taught at the TU/e.

3.1 Syntax

Predicate logic.

3.2 Semantics

State transformations.

3.3 Mutex properties

One of the properties of the system, ME, can be expressed; EA cannot.

Properties of the schedular cannot be expressed either.

3.4 Assessment

3.4.1 The matrix

Hoare formulas are the syntax for specification of properties. System structure is used to determine where assertions hold.

	syntax	semantics	abstraction	compositionality	automa
System	programming language	state transformation/logical truth	very little	yes	see lat
Connection	Hoare triple	logical implication	-	-	see lat
Specification	predicate formulas	logical truth	very little	yes	see lat

The properties are specified about parts of the system, i.e., compositionally.

3.4.2 The V's

Hoare specifications of properties fit in the V at the upper three boxes where it acts as specifications of properties of the system but also at the opposite boxes as the tests may be specified in this formalism.

Chapter 4

Specification: Linear Temporal Logic (LTL) - global

We present a syntax and semantics for property specifications.

DP.5: Introductory remarks chapter 5, 5.1, 5.2, 5.4.

NB DP.5.3 is addresed in the next chapter, **??**.

4.1 Syntax

DP.5.4 Syntax: propositional or predicate logic extended with temporal operators.

4.2 Semantics

DP.5.4 Semantics: non-temporal formulas interpreted on states; temporal symbols interpreted over state sequences.

NB State sequences are the semantics of transition systems as well, so there is a connection to the semantics of systems.

Check whether you are familiar with the contents of DP chapter 2 and 3 as far as needed for understanding the formal definition of the semantics.

4.3 Mutex properties

Crucial is DP.125.

The properties of the system, ME and EA, can be expressed formally in LTL.

	syntax	semantics	abstraction	compositionality	automation
System	-	-	-	-	-
Connection	-	-	-	-	-
Specification	LTL formulas	set state seq's	no system structure	no	-

The properties of the scheduler, required to make the properties of the system hold, can also be expressed formally in LTL.

The required schedular properties can be put to the system as well as to the algorithm (as a premisse for good behaviour), depending on who has the responsibility for the schedular and algorithm match.

4.4 Assessment

4.4.1 The matrix

LTL formulas are the syntax for specification of properties, sets of state sequences the semantics. System structure is abstracted away. Because the logic specifies order of state changes, it should also be noted that this order is time order, but abstracts away real time. The properties are specified about the whole system, i.e., global.

4.4.2 The V's

LTL specifications of properties fit in the V at the upper three boxes where it acts as specifications of properties of the system but also at the opposite boxes as the tests may be specified in this formalism.

Chapter 5

Verification: CTL ModelCheck - global - semantic - automated

We present a model check procedure to prove properties, expressed in temporal logic, of systems, modelled as transition systems. We deviate from DP. in using Computation Tree Logic (CTL), a branching time version of temporal logic rather than the linear time LTL from chapter 4. See M.R.A. Huth and M.D. Ryan, Cambridge University Press, 2004 (2000) - copies of the relevant chapter are provided at the lectures.

5.1 System satisfying property - semantical

A system P, represented as a transition system, has as semantics a set of state sequences, say $\|[P]\|$.

A property S, specified as a temporal logic formula, is true or false for a sequence; the property also detemines a set of state sequences: those for which it is true, say $\|[S]\|$.

A system P satisfies a specification S if $\|[P]\| \subseteq \|[S]\|$.

5.2 System satisfying property - ModelCheck

In the previous chapter, **??**, we have seen how at the syntact level properties about systems can be proven. We also defined what it meant semantically for a property to hold for a system: inclusion of state sequence sets. Automation of verification can be done at both the semantic and the syntactic level. In the present chapter we present automation of semantic verification, in chapter 9 we present an automation of a syntactic approach.

To reason about set inclusion between sets of sequences is awkward: usually the sequences are infinitely long and there are infinitely many of them. The first idea is, to translate set inclusion between sets of sequences to a check between finite representations of these sets of sequences. The second idea is, to automate this check.

1. The finite transition system (syntactically) representing the system is unfolded into one with concrete states: the (semantic) model.
2. The temporal formula (syntactically) representing the property is checked as follows.

Peled, Springer A finite transition system with concrete states, i.e., a model, for the negation of the formula is constructed. In this case model checking means automata theoretic checking the intersection of the models for emptiness (then the property holds for the system).
Disadvantage: constructing the model for the negation of the formula is difficult.
Advantage: this works for LTL.

Huth and Ryan, OUP The formula is checked on the model incrementally, starting with the smallest subformulas.
Disadvantage: this only works for CTL.
Advantage: the model checking is easy and intuitive.

Note, that although we call this a semantical approach, we stay in fact, especially in the second case, quite close to the syntax. The reason to still call it a semantical approach is that the model check algorithm is quite directly using the semantics of the transition system: the idea of sequences of states.

Latest news: The intuitive approach of CTL checking can, after all, be extended to LTL as well (and even CTL). Kesten and Pnueli, to appear in Elsevier Science.

We discuss Huth and Ryan NB These parts of Huth and Ryan have been provided as handouts at the lectures:

1. CTL HR.152-159;
2. The model checking procedure HR.172-174;
3. Exemplified for eventual access on Mutex (2 versions) HR.177.

5.3 Assessment

5.3.1 The matrix

For a system the syntax is a transition system, the semantics is a tree. For a specification of a property, the syntax is CTL, the semantics a set of trees. The connection between the two is, at the semantic level, the element-of relation. On the modelcheck level the syntactic, CTL, formula is put to the syntactic, transition system, representation of the system. Nevertheless this is regarded as a semantic approach, because the algorithm that performs the modewl check interprets the transition system in a quite semantical fashion, namely using the transitions through the tree to determine which formulas get attached to its nodes. The representation of the system is global, so the model check approach itself is global as well - alternative, compositional versions exist but are beyond the scope of the lectures.

	syntax	semantics	abstraction	compositionality	automation
System	trans. system	tree state seq's	no system structure	-	-
Connection	model check	element-of	-	no	yes
Specification	CTL formulas	set tree stat seq's	no system structure	-	-

5.3.2 The V's

CTL automated verification fits in the V as properties in the right hand part being checked against models of the system in the lefthand part.

Chapter 6

Specification and verification: Hoare logic

A syntax and semantics for property specifications for systems and properties can be found in DP 7, notably DP.7.4: We assume familiarity with Hoare logic as taught at the TU/e.

6.1 Syntax

Predicate logic.

6.2 Semantics

State transformations.

6.3 Mutex properties

One of the properties of the system, ME, can be expressed; EA cannot.

Properties of the schedular cannot be expressed either.

6.4 Assessment

6.4.1 The matrix

Hoare formulas are the syntax for specification of properties. System structure is used to determine where assertions hold.

	syntax	semantics	abstraction	compositionality	automa
System	programming language	state transformation/logical truth	very little	yes	see lat
Connection	Hoare triple	logical implication	-	-	see lat
Specification	predicate formulas	logical truth	very little	yes	see lat

The properties are specified about parts of the system, i.e., compositionally.

6.4.2 The V’s

Hoare specifications of properties fit in the V at the upper three boxes where it acts as specifications of properties of the system but also at the opposite boxes as the tests may be specified in this formalism.

automated

Chapter 7

Process Algebra

At this point in the lecture series we have introduced the concepts of formal methods, mainly using temporal formalisms.

We now introduce not so much new concepts as well as alternatives for the formalizations encountered so far. Process algebra is a case in point: from a high vantage point it can be viewed as being a specification and verification formalism in much the same way as the ones encountered so far.

However, from closer up it is quite different, as will show up in the assessment matrix.

We present process algebra along the lines of DP.

DP.8: Introductory remarks chapter 8, 8.1, 8.2, 8.3 (especially the idea that state variable values are modelled as processes: hence the three additional processes in the Dekker version of mutual exclusion), 8.5, 8.6.

(At least) process algebra is quite special because of the following features:

1. The system model and the property model are one and the same, not only the semantics, but also the syntax.
2. The level, of abstraction is higher than usual: actions rather than state variable values.
3. The relation between system and specification is not so much stating that a property holds of a system as well as stating that one system refines another.
4. The formal proof system is a rewrite system rather than a logic.

It is especially because of the last two points that, to accomodate for several notions of refinement, various notions of equivalences between agents have been defined. Furthermore, to enable establishing refinement between agent that differ only in, intuitively, internal behaviour, a powerful notion of hiding and silent steps are part of the approach.

	syntax	semantics	abstraction	compositionality	automation
System	agent	action tree	no state information	yes	-
Connection	rewrite rules	equivalences	-	-	-
Specification	agent	action tree	no state information	yes	-

7.1 Assessment

7.1.1 The matrix

For systems as well as properties, the syntax is an agent, a process algebra term, the semantics an action tree. The connection between the two is, at the semantic level, various notions of simulation. On the proofsystem level, equivalence is shown by rewriting.

Various possibilities for automation are available but outside the scope of the lectures

7.1.2 The V's

Process algebra fits in the V at the higher parts of the left branch, i.e., for specification and also on the higher parts of the connection between the left and right branch, i.e., for testing. Because of the high abstraction level the scope is more towards protocols than towards large systems.

Chapter 8

Testing

DP.9: Introductory remarks chapter 9, 9.1, 9.2, 9.3, 9.8.

This chapter is different from the previous ones. It is not A test formalism that is introduced here, but rather a collection of formal notions ABOUT testing that enable to more rigorously describe PROPERTIES of test formalisms.

We therefore do not asses a formalism, but rather testing as such.

8.1 Assessment

8.1.1 The matrix

For systems the syntax is transition systems, the semantics is state sequences. For specification of properties, the syntax is LTL, the semantics state sequences. The connection between the two is, at the semantic level, set inclusion of sets of state sequences. On the proofsystem level the state sequences do not appear anymore: the proofs are given in terms of the transition systems and the proof rules.

8.1.2 The V's

Testing fits in the V as the connections between the left and right branches of the V.

	syntax	semantics	abstraction	compositionality	automation
System	code or a model	execution or model	various	possibly	yes
Connection	various	various	various	-	possibly
Specification	various	execution related	various	possibly	-

Chapter 9

Verification: Hoare OO Theorem Prover - compositional - syntactic - automated

See power point of Huizing. Note, that this completes the formal methods picture sketched in the lecture series as it adds automation of the syntax-oriented proof approach through the ESC/Java prooftool.

Chapter 10

Verification at Imtech

As in the not-for-public-disclosure powerpoint presentation at the lecture following Van Gerwen - imTech.