Lecture 1: Introduction
and analysis with Alloy
and declarative modelling
micromodels of software

Marktoberdorf, August 2002
MIT Lab for Computer Science
Daniel Jackson
lightweight models
lightweight models

a foundation for robust, usable programs

lightweight models
lightweight models

- elements
- small & simple notations
- partial models & analyses
- full automation

a foundation for robust, usable programs
Lightweight models

lightweight models

a foundation for robust, usable programs

focus on risky aspects

high cost of failure

structure-determining

hard to get right or to check

small & simple notations

full automation

partial models & analyses

Elements

Lightweight models
what assurance costs
What assurance costs
What assurance costs
what assurance costs
what assurance costs
What assurance costs
what assurance costs

- type-checked models
- write-only models
- sketching
- hacking

assurance

cost
what assurance costs
what assurance costs
my work in marktoberdorf context
my work in marktobberdorf context
my work in marktoberdorfter context

> simulation, not just checking
> like Harel, not Rushby & Moore
> complementary to Harel & Pnueli
> relational, not algebraic (cf. Tarlecki and Meseguer)
> designed for experts, but not super-experts
> underlying idioms due to Hoare, Woodcock et al.
my work in marktoberdorf context
started this in 1994, and have had some successes but much less mature than ACL2, PVS, Statemate, etc.

my work in marktoberdorf context

role of mathematics

> only way to make things simple
> semantics in terms of sets, and SAT

prepared for experts, but not super-experts like Harel, not Rushby & Moore

complementary to Harel & Pnueli

relation, not algebraic (cf. Tarlecki and Meseguer)

simulation, not just checking

> underlying idioms due to Hoare, Woodcock et al.
features of Alloy
features of Alloy

› express complex structure, static and dynamic structural

› with just a few powerful operators
Features of Alloy

dependent

- structural

- describe system as collection of constraints
- a full logic, with conjunction and negation
- declarative

- with just a few powerful operators
- express complex structure, static and dynamic
Features of Alloy

- Fully automatable
- Simulational & checking
- Analyzable
- Describe system as collection of constraints
- A full logic, with conjunction and negation
- Declarative
- With just a few powerful operators
- Express complex structure, static and dynamic
- Structural
structure is everywhere

... library catalogs, address books, phone networks, highway systems, postal routes, company organizations, ...
structural

structure is everywhere
› highway systems, postal routes, company organizations, library catalogues, address books, phone networks, ...

structure is becoming more pervasive
› self-assembling software (eg, Observer pattern)
› memory gets cheaper: address books in every phone
tool researchers have neglected structure

structure is becoming more pervasive

library catalogs, address books, phone networks, "highway systems, postal routes, company organizations, ...

...
structure is everywhere

There is no problem in computer science that cannot be solved by introducing another level of indirection, but that usually reveals new problems.

Tool researchers have neglected structure.

David Wheeler

...
declarative
declarative

declarative description
› model is collection of properties
› the more you say, the less happens
Sys meets Prop: $\text{Sys} \Rightarrow \text{Prop}$

Advantages

- **Simplicity**: no separate language of properties
- **Partiality**: doesn’t require special constructs
- **Incrementality**: to say more, add a property
- **The more you say, the less happens**

Declerative description

- model is collection of properties
Less constrained environment means greater freedom.

\[ \text{Sys meets Prop: } \text{Sys} \Rightarrow \text{Prop} \]

Why less is more:

- **Simplicity:** no separate language of properties
- **Partiality:** doesn't require special constructs
- **Incrementality:** to say more, add a property

advantages

- the more you say, the less happens
- model is collection of properties
- declarative description

declarative
analyzable
analyzable

but missed opportunity (and wishful thinking)
analyze

analyzable

write-only models

useful if precise enough

write-only models

but missed opportunity (and wishful thinking)

tool-assisted modelling

simulate and check incrementally

catch errors early, develop confidence

optimize for failing case: most of my examples will be wrong

optimize for failing case: most of my examples will be wrong
analyzable

Unlike testing, billions cases/second
Like testing, sound but not complete
Concrete: Generates samples & counterexamples
Fully automatic, with no user intervention
Alloy's analysis

Optimize for failing case: most of my examples will be wrong
Catch errors early, develop confidence
Simulate and check incrementally
Tool-assisted modelling

But missed opportunity (and wishful thinking)
Useful if precise enough
Write-only models
analyzable
Small Tower of 6 Gears, Arthur Ganson

declarative & executable?
Small Tower of 6 Gears, Arthur Ganson

Traditionally, good arguments for both declarative XOR executable are executable.
declarative & executable?

traditionally declarative XOR executable

but can have cake and eat it

Small Tower of 6 Gears, Arthur Ganson

with right analysis technology

Good arguments for both declarative & executable

traditionally
declarative & executable?

traditionally declarative XOR executable

but can have cake and eat it

Alloy’s analysis can execute a model forwards or backwards

without test cases

no ad hoc restrictions on logic

with right analysis technology

Small Tower of 6 Gears, Arthur Ganson
a numbering problem
given document whose paragraphs are tagged with styles
style sheet that gives numbering rules for styles

a numbering problem
A numbering problem

Given a document whose paragraphs are tagged with styles, produce a style sheet that gives numbering rules for styles like my Marktoberdorf notes.
I like my Marktoberdorf notes:

- Document with numbered paragraphs
- Style sheet that gives numbering rules for styles
- Given a numbering problem
Given a numbering problem, a document whose paragraphs are tagged with styles produce a style sheet that gives numbering rules for styles. A document with numbered paragraphs (like my Marktoberdorf notes) gives a numbering problem.
a candidate solution

• optionally, a parent
• an initial value for numbering

style sheet assigns to each style
A candidate solution
a candidate solution
a candidate solution

Part A: Introduction
A.1 Motivation
A.1.1 Why?
A.2 Overview

Part B: Conclusions
B.1 Unrelated Work

Part: Conclusions
section
subsection
part
section
subsection
declare styles & parent relation

sig Style {parent: option Style}
declarer styles & parent relation

styles
declare styles & parent relation

sig Style {parent: Style}

run Show

run Show ()

ask for a sample

decide styles & parent relation

styles
declare styles & parent relation

{style Style {parent: some parent}}

run show

run show ()

ask for a sample

sig Style {parent: option Style}

declare styles & parent relation

styles
\{ \textbf{fact}\} \texttt{Acyclic} (parent)

constraint parent relation to be acyclic

\texttt{run Show}

\texttt{run Show () some parent}

ask for a sample

\texttt{sig Style \{ parent: option Style \}}

declare styles & parent relation

\texttt{styles}
\{ \text{fact} \ \text{acyclic (parent)} \}

constraint parent relation to be acyclic

run Show

run Show \{ \text{some parent} \}

don't ask for a sample

\{ \text{sig} \ \text{Style option: parent} \}

declare styles as parent relation

styles
fun Acyclic [t] (r: t -> t) {no iden[t] & ^r} 

how to define acyclic

{fact Acyclic (parent)}

constraint parent relation to be acyclic

run Show

{run Show ()} 

ask for a sample

sig Style {parent: Style}

declarer styles & parent relation

styles
numbers

introduce numbers

{this != next} {next: option Number

sig Number

introduce numbers

numbers
add numbers to styles

{next; this = next}

introduce numbers

numbers

{fact} Style = NumberedStyle

{sig NumberedStyle extends Style {init: Number}

sig Number {next: Option Number

} Number

sig numbers
introduce numbers

sig

Number {next: Option Number}

this != next

add numbers to styles

fact

Style = Number

NumberedStyle extends Style {init: Number}

add numbers to styles

run

Show (some parent)

fun

Show ()

ask for a sample

fun

Show

{some parent}

run

Show ()

run

Show

{some parent}

run

Show ()

run

Show
13 numbers introduce numbers sig Number {next: option Number}

add numbers to styles

{fact Style = NumberedStyle}
{sig NumberedStyle extends Style {init: Number}}

add numbers to styles

{this} {next: option Number}

introduce numbers
declare numbering

sig Numbering

{num: Style ->? Number}

declarer numbering

numbering
for 2 but 1 numbering
run ShowNumbering
{some num} run ShowNumbering ()
ask for a sample

{num: Style -> Number}

sig Numbering
declare numbering

numNumbering
Numbering {num: Style -> Number}

fun ShowNumbering () {
  {num | some num |}{num: Style => Number |}
}

dec {num: Style => Number |}

declare numbering

for 2 but 1 numbering
run ShowNumbering
{num ShowNumbering () |some num |}
ask for a sample
Numbering

Numbering

for 2 but 1 Numbering
run ShowNumbering
{run ShowNumbering () {
  ask for a sample

  {num: Style => ? Number
    Numbering
  }
}
]
declare Numbering

Numbering

Numbering
numbering algorithm
numbering algorithm

what numbering

for paragraph of style \(s\)? i.e., just gave numbering \(n\)

must now generate numbering \(n\)

encounter paragraph with style \(s\)

i.e., just gave numbering \(n\)

what numbering \(n\) follows \(n\) for paragraph of style \(s\)?

numbëring algorithm
numbering algorithm

what numbering follows for paragraph of style s? 

\[ \text{fun Next (n,n': Numbering, s: Style) } \]

an attempt:

- must now generate numbering n
- encounter paragraph with style s
  - if just gave numbering n
  - what numbering follows n for paragraph of style s?

numbering algorithm
run Next for 3 but 2 Numbering showing next next next
grandchild styles is numbered with initial value

run Next for 3 but 2 Numbering

showing next
guiding the simulation
guiding the simulation

fun ShowNext (n,n': Numbering, s: Style) {
  Next (n,n',s) && some n.num[s..parent]
}

run ShowNext for 3 but 2 Numbering
fun ShowNext (n, n': Numbering, s: Style) { Next (n, n', s) & some n.num[s~parent] } run ShowNext for 3 but 2 Numbering

guiding the simulation
ShowNext \( n, n': \text{Numbering}, s: \text{Style} \) \{ 
Next \( n, n', s \) \&
\text{some } n'.num[\text{num}'s\text{-parent}] 
\}

run ShowNext \( 3 \text{ but 2 Numbering} \)

Guiding the simulation
guiding the simulation

run ShowNext 3 but 2 numbering
{Next (n, n', s) as some n.num~parent }
run ShowNext (n, n', Numbering, s: Style) (n, n', s: Style) because no next!
 loses its number root style s

run ShowNext for 3 but 2 numbering

run ShowNext (n, n', Numbering, s: Style)
Fixing the operation
fixing the operation
focusing the operation
```javascript
Next (n, n': Numbering, s: Style) {
  let i = n.num[s] |
  some i =>
    some i.next
  n'.num = {d: s.^parent, x: Number | x = n.num[d]} + s ->
  if no n.num[s] then s.init else n.num[s].next
}
```

Fixing the operation
fixing the operation

Next (n, n': Numbering, s: Style) {
  let i = n.num[s].next |
  some i =>
  some i.next

  if no n.num[s].next
  then s.init
  else n.num[s].next

  s.init

  style s gets

  Initial value
  numbered within
  style s gets

  Tun Next (n', n: Numbering, s: Style)
guiding the simulation
run ShowNext for 3 but 2 Numbering

fun ShowNext (n,n': Numbering, s: Style) {
    Next (n,n',s)
    &&
    some n.num[~parent]
    &&
    some n.num[s]
}

fun ShowNext (n,n': Numbering, s: Style) {
    Next (n,n',s)
    &&
    some n.num[~parent]
    &&
    some n.num[s]
}

Guiding the simulation
run ShowNext for 3 but 2 Numbering
{ Next (n, n′) if some n.num = parent ~ some n′.num }
run ShowNext (n′: Numbering, s: Style)

guiding the simulation
run ShowNext for 3 but 2 Numbering

{ Next (n.n', s) \&\& some n.n'[s] \&\& some n.n'.parent
}

run ShowNext (n.n': Numbering, s: Style)

guiding the simulation
checking a property
checking a property

if style is not a parent, step is reversible

assert Reversible

{ Next(n0,n's) => n0.num = n1.num |
  all no, n1, n: Numbering, s: Style - Style.parent
  } assert Reversible

if style is not a parent, step is reversible

checking a property
checking a property

if style is not a parent, step is reversible

assert Reversible

Reversible { all no, n: Numbering, s: Style - Style::parent |
  Next(n0, n.s) => n0.num = n1.num
} assert Reversible

if style is not a parent, step is reversible

checking a property
trying again...
trying again...

\text{fact \{Injective\,next\}}

make numbering injective

trying again...
trying again…

\texttt{\textbf{fact} \{Injective (next)\}}

make numbering injective

does this fix the problem?

trying again…
counterexample
example after numbering $n$
after numbering n

counterexample
counterexample

"after numbering n

adjacent style has no number afterwards"
In and on numbers before counterexample, ctd
counterexample, ctd

In 0 and n

numberings

before
counterexample, ctd
counterexample, ctd

No adjacent styles numbers before

No and in

Before numberings

Before

Value eliminated, so value before was irrelevant!
check ReversibleWhenLine

assert

check ReversibleWhenLine
counterexample, again
counterexample, again
counterexample, again
counterexample, again
Counterexample, again:

Is confused! Theorem: initialization and increment are distinct! Theorem is confused!
checking a refactoring
are these equivalent?

does this work?

checking a refactoring
checking a refactoring

are these equivalent?

fun `Next1 (n,n': Numbering, s: Style) {
    n'.num = {d: s.^parent, x: Number | x = n.num[d]} + s
    if no n.num[s] then s.init else n.num[s].next
}

fun `Next2 (n,n': Numbering, s: Style) {
    all d: s.^parent | n'.num[d] = n.num[d] n'.num[s] =
    if no n.num[s] then s.init else n.num[s].next
}

assert Same {
    all n,n': Numbering, s: Style | `Next1(n,n',s) \iff `Next2(n,n',s)
}

ask the tool:

{ if no n.num[s] then s.init else n.num[s].next
}

are these equivalent?
what happened
What happened

everyone focuses on small models
- avoids wasted time
- to get key properties
- constrains just enough
- write a bit, analyze a bit

incrementally
What happened

- Analyze small models
- Encourages small models
- Avoids wasted time,
- To get key properties
- Constraint just enough
- Write a bit, analyze a bit
- Incrementality

What happened

- Style hierarchy a tree? Line?
- Two numbers have same next?
- Number must have next?
- Avoids wasted time, encourages small models
Two numbers have same next?
Number must have next?
Analyses prompted questions
Encourages small models
Avoids wasted time
To get key properties
Constraint just enough
Write a bit, analyze a bit
Incrementality

What happened
declarative vs. operational development

- declarative: no behaviours; satisfies all safety properties
- operational: satisfies no safety properties; all behaviours; satisfies a safety property

Operational vs. declarative development
What's been done?
What’s been done?

- Transaction cache (Tucker)
- Chord peer-to-peer lookup (Wee)
- Intentional naming (Khurshid)

Analyzing implemented systems
Analyzing existing models

- Classic distributed algorithms (Shlyakhter, from SWV)
- UML meta model (Vaziri, from OCL)
- Unison file synchronizer (Nolte, from Pierce's maths)
- Firewire leader election (me, from Vaandrager's IOA)
- Microsoft COM (Sulliven, from Z)
- Transaction cache (Tucker)
- Chord peer-to-peer lookup (Whee)
- Intentional naming (Khursheed)

What's been done?
What's been done?

- 200 lines of Alloy, 30-200 hours work

Typically:

- Classic distributed algorithms (Shlyakhter, from SMV)
- UML meta model (Vaziri, from OCL)
- Unison file synchronizer (Nolte, from Pierce’s mathis)
- Firewire leader election (me, from Vandrager’s IOA)
- Microsoft COM (Sullivan, from Z)
- Analyzing existing models
- Transaction cache (Tucker)
- Chord peer-to-peer lookup (Whee)
- Intentional naming (Khurshid)

Analyzing implemented systems
example: intentional naming
example: intentional naming

query scheme

result of query is set of simple names

intentional names are trees
Example: Intentional naming

Result of query is set of simple names

Intentional names are trees

Query scheme
example: intentional naming

query scheme

- intentional names are trees
- result of query is set of simple names
Example: Intentional naming

- Intentional names are trees
- Result of query is set of simple names
- Query scheme

```
query

Building

Service

Camera

Printer

Database
```
results
This algorithm uses the assumption that omitted at
tributes correspond to wild-cards; this is fine for both queries
and adversitements. A nice property of the algorithm is that

Results
what we did

This algorithm uses the assumption that omitted at

Results

and adverseriments. A nice property of the algorithm is that

tributes correspond to wild-cards; this is true for both queries
what we did — analyzed claims made in paper: mostly untrue

Results
what we did

- analyzed algebraic properties: also untrue
- analyzed claims made in paper: mostly untrue

It is monotonic

and advertisements. A nice property of the algorithm is that

This algorithm uses the assumption that omitted at
adapted model for fixes in code: also broken

\( \text{and is monotonic} \)

analyzed algebraic properties: also untrue

analyzed claims made in paper: mostly untrue

what we did

Results

\text{A nice property of the algorithm is that this corresponds to wild-cards; this is true for both queries. This algorithm uses the assumption that omitted a-}
What we did: analyzed new semantics & checked it. Developed new model for fixes in code; also broken. Adapted model for fixes in code; also broken. Developed new semantics & checked it.

Results:
and advertisements. A nice property of the algorithm is that

tributes correspond to wild-cards; this is true for both queries.
This algorithm uses the assumption that omitted at
what we did:
- analyzed claims made in paper: mostly untrue
- analyzed algebraic properties: also untrue
- adapted model for fixes in code: also broken
- `add` is monotonic
- developed new semantics & checked it

Reflections

- This algorithm uses the assumption that omitted at
- This matches corresponding to wild-cards; this is true for both queries.
- A nice property of the algorithm is that
- and advertisements
What we did

- initial analyses took 2 weeks and 100 lines of Alloy

Reflections

- developed new semantics & checked it
- adapted model for fixes in code: also broken
- eg, add is monotonic
- algebraic properties: also untrue
- analyzed claims made in paper: mostly untrue

Results

and advertisements. A nice property of the algorithm is that
 This algorithm uses the assumption that omitted al
Reflections

- Developed new semantics & checked it
- Analyzed algebraic properties: also untrue

What we did

- Found all bugs in trees of 4 nodes or less -- approx 10 secs
- Initial analysis took 2 weeks and 100 lines of Alloy

Results
31 results
what we did›analyzed claims made in paper: mostly untrue
›analyzed algebraic properties: also untrue e.g., add is monotonic
andalryzed algebraic properties: also untrue
what we did

2000 lines of tests hadn't found bugs in a year
Found all bugs in trees of 4 nodes or less -- approx 10 secs
initial analysis took 2 weeks and 100 lines of Alloy

Reflections
developed new semantics & checked it
adapted model for fixes in code: also broken
eg, add is monotonic
analryzed claims made in paper: mostly untrue
challenge: get numbering right
challenge: get numbering right

section has parent chapter and appendix

multiple parents

section and figure have parent chapter

multiple children

fix the numbering mechanism to handle
What is a model?
what is a model?

a representation of a system

› more or less useful, not more or less correct [Fowler]
› useful to the extent that it answers questions [Ross]
What is a model?

- A representation of a system, to be built
- To explore design of system to be built
- To explain & evaluate existing system
- Role of a model

[Useful to the extent that it answers questions [Ross]]
- More or less useful, not more or less correct [Fowler]
why model?
why model?

'plan to throw one away' [Brooks]

nasty surprises happen sooner

'100 line model or 100k lines of code?'
why model?

- allow delegation & division of labour
- easier to use and implement
- designs with clear conceptual models
- nasty surprises happen sooner
- 100 line model or 100k lines of code?

plan to throw one away? [Brooks]

why model?
why model?

- not a good use of testing
- conceptual flaws get mired in code
- separation of concerns
- allow delegation & division of labour
- easier to use and implement
designs with clear conceptual models
- nasty surprises happen sooner sooner
- 100 line model or 100K lines of code?
- planning to throw one away? [Brooks]
lightweight formal methods
lightweight formal methods

- full automation
- partial models & analyses
- small & simple notations

Elements
High weight formal methods

- Lightweight formal methods
  - Small & simple notations
  - Partial models & analyses
  - Full automation
  - Structure-determining
  - Focus on risky aspects
  - Hard to get right, or to check
  - High cost of failure