System Modelling and Design
COMP2111

Johannes Åman Pohjola

+ tutors:
Zhuo (Zoey) Chen
Raphael Douglas Giles

Formal

System Modelling and Design COMP2111

Johannes Åman Pohjola

+ tutors:
  Zhuo (Zoey) Chen
  Raphael Douglas Giles

Formal

System Modelling and Design
COMP2111

Credit for the material also goes to:
Paul Hunter,
Christine Rizkallah,
Liam O'Connor,
and Carroll Morgan

Johannes Åman Pohjola

+ tutors:
Zhuo (Zoey) Chen
Raphael Douglas Giles

We’ll learn to

model systems in a way that’s unambiguous and mathematically precise.

We’ll be able to

say what it means for a system to satisfy its specification,

and prove that it does so.

We’ll need

a substantial toolbox of discrete math and formal logic.

Don’t worry; we’ll teach it, not assume it.

Non-examples

September 1981

Transmission Control Protocol
Functional Specification

3.9. Event Processing

The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance.

The activity of the TCP can be characterized as responding to events. The events that occur can be cast into three categories: user calls, arriving segments, and timeouts. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection.

Events that occur:

User Calls

- OPEN
- SEND
- RECEIVE
- CLOSE
- ABORT
- STATUS

Arriving Segments

- SEGMENT ARRIVES

Timeouts

- USER TIMEOUT
- RETRANSMISSION TIMEOUT
- TIME-WAIT TIMEOUT

The model of the TCP/user interface is that user commands receive an immediate return and possibly a delayed response via an event or pseudo interrupt. In the following descriptions, the term "signal" means cause a delayed response.

Error responses are given as character strings. For example, user commands referencing connections that do not exist receive "error: connection not open".

Please note in the following that all arithmetic on sequence numbers, acknowledgment numbers, windows, et cetera, is modulo 2**32 the size of the sequence number space. Also note that "=<" means less than or equal to (modulo 2**32).

Non-examples

This RFC is a specification in English.

Natural language specs tend to have:

- Ambiguities
- Room for interpretation
- Important details in the writer's head absent from actual text.

3.9. Event Processing

The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance.

The activity of the TCP can be characterized as responding to events. The events that occur can be cast into three categories: user calls, arriving segments, and timeouts. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection.

Events that occur:

User Calls

OPEN
SEND
RECEIVE
CLOSE
ABORT
STATUS

Arriving Segments

SEGMENT ARRIVES

Timeouts

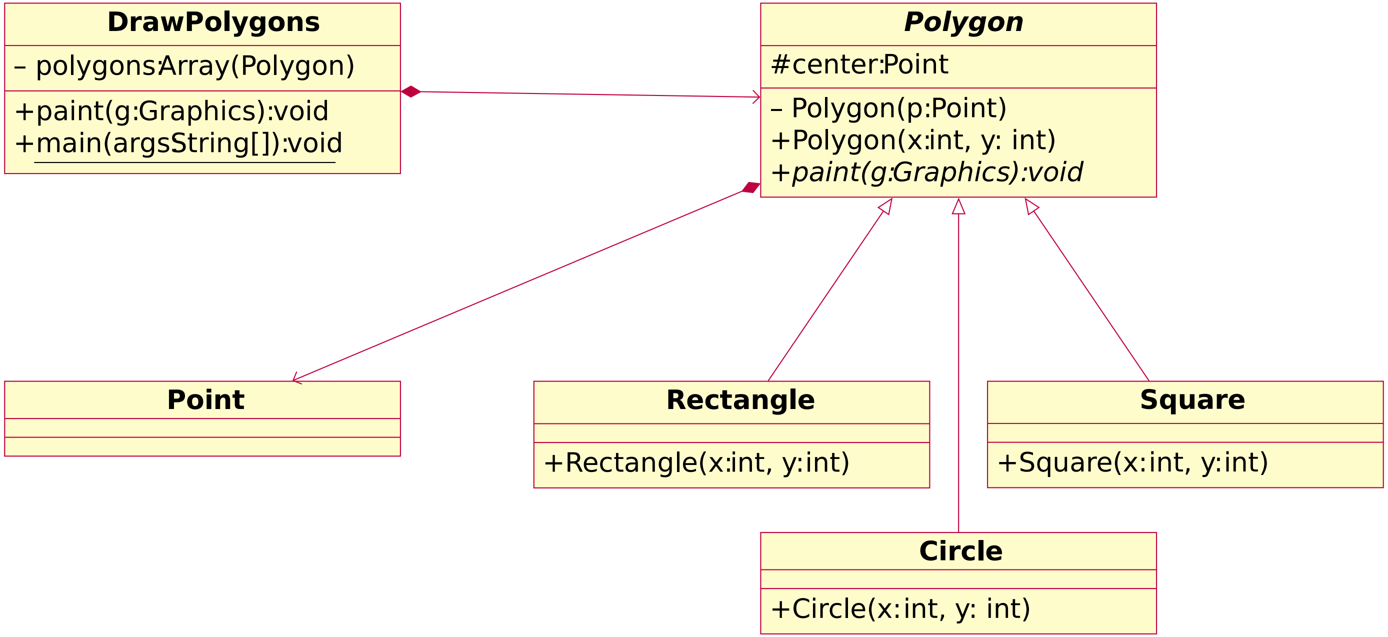
USER TIMEOUT
RETRANSMISSION TIMEOUT
TIME-WAIT TIMEOUT

The model of the TCP/user interface is that user commands receive an immediate return and possibly a delayed response via an event or pseudo interrupt. In the following descriptions, the term "signal" means cause a delayed response.

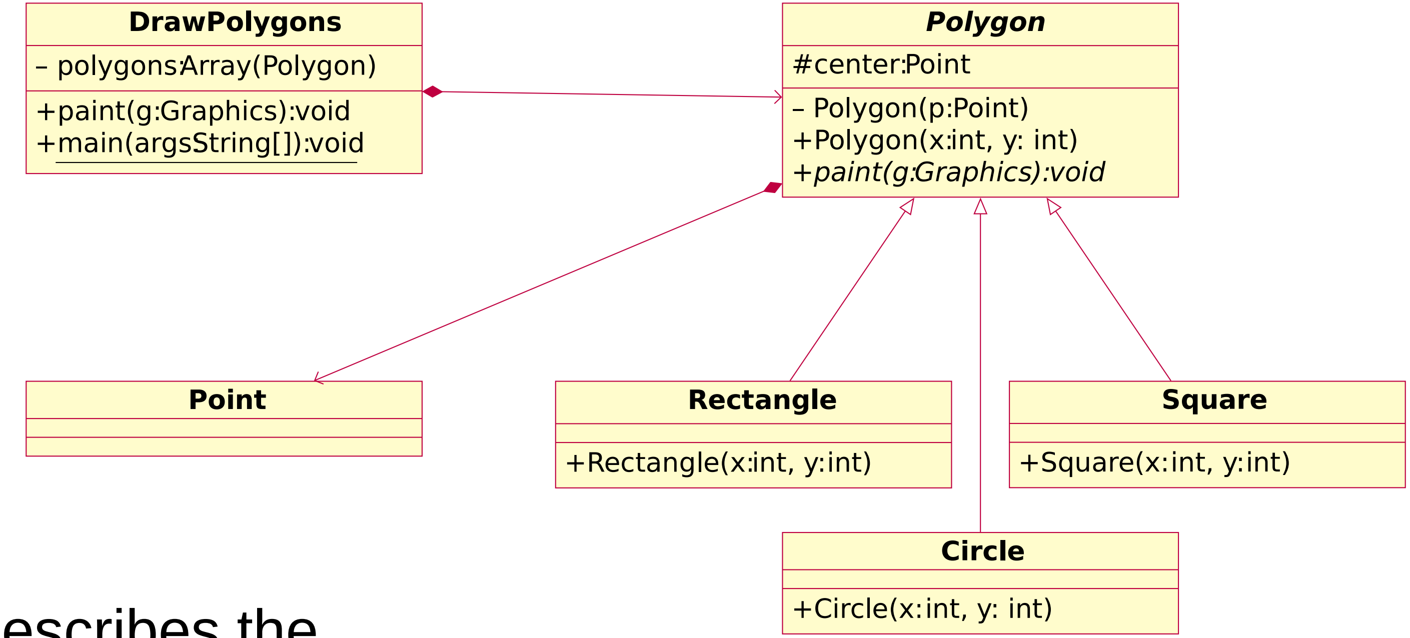
Error responses are given as character strings. For example, user commands referencing connections that do not exist receive "error: connection not open".

Please note in the following that all arithmetic on sequence numbers, acknowledgment numbers, windows, et cetera, is modulo 2**32 the size of the sequence number space. Also note that "=<" means less than or equal to (modulo 2**32).

Non-examples



Non-examples



This UML diagram describes the *structure* of the system, not its behaviour.

Resources

Course website: http://www.cse.unsw.edu.au/~cs2111

- Lecture slides, tutorials
- Assignment instructions
- ...

Ed forum: https://edstem.org/au/courses/15105/

- General announcements
- Class discussion, announcements
- E-mail cs2111@cse.unsw.edu.au if you haven't been invited!

Moodle: https://moodle.telt.unsw.edu.au/

- Lecture recordings
- Weekly quizzes

Examination

- Weekly quizzes: 15 credits total
  - After the lectures of *every* week (except W6 and W10).
  - Will appear on Moodle.
  - Deadline: Monday 4PM (before start of next week’s lectures)

- Three assignments (individual or pair, written): 11+12+12=35 credits

- Final exam (online, format TBA): 50 credits

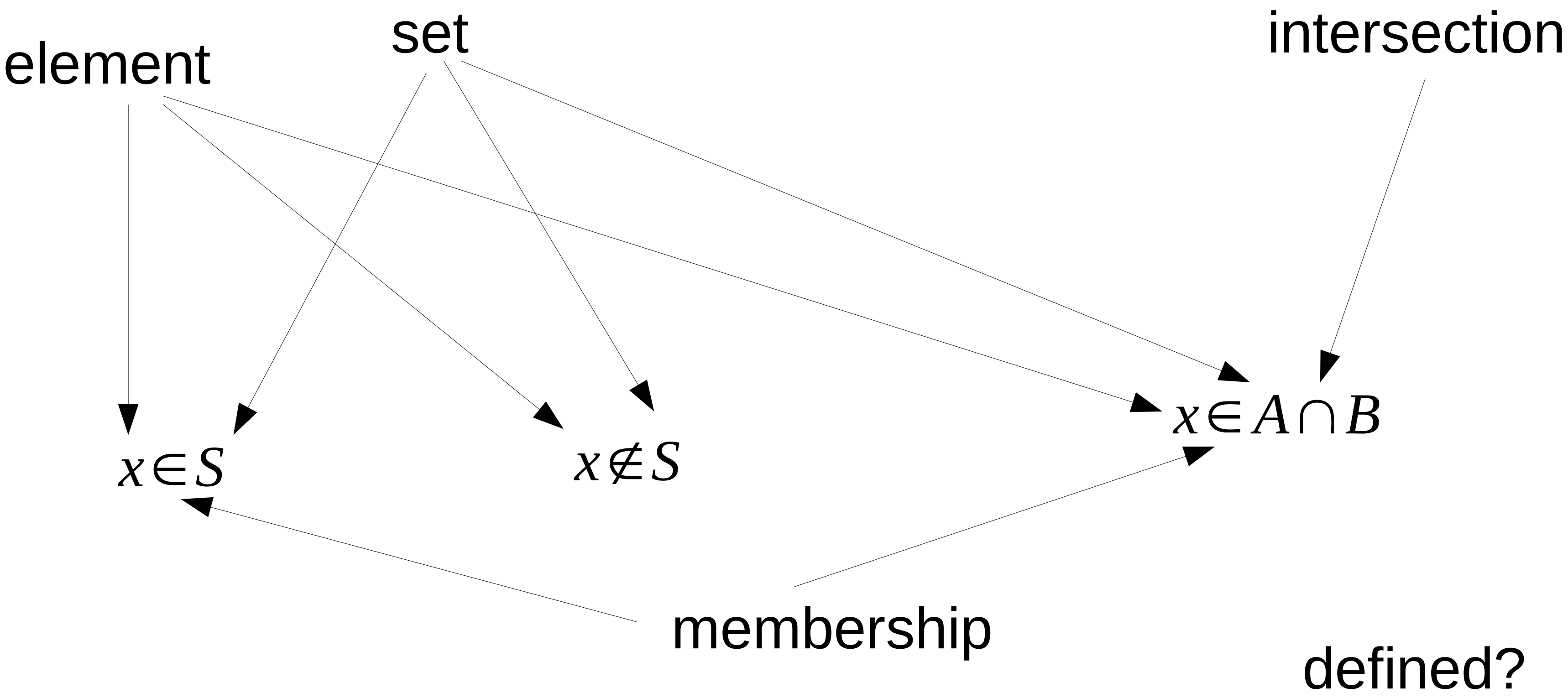
Introduction to “Formal” Logic

Start here → D.4 Calculating with conditions

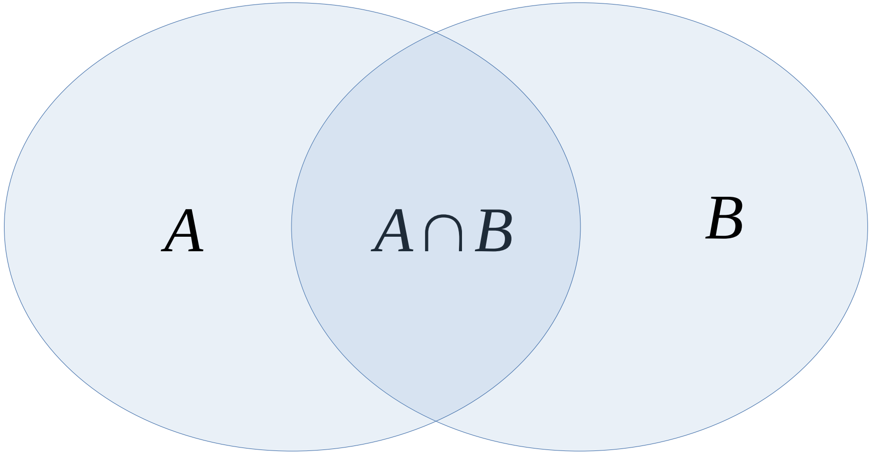
This is a (draft) textbook for COMP6721 (In-)Formal Methods by Carroll Morgan

It’s on the course website.

Introduction to “Formal” Logic



Introduction to “Formal” Logic



$x \in A \cap B$ if and only if $x \in A$ and $x \in B$

Introduction to "Formal" Logic

intersection	$A \cap B$
union	$A \cup B$
subset	$A \subseteq B$

Q: Subset is not like others in an important way. How?

$x \in A \cup B$	if and only if	$x \in A$	or	$x \in B$
$A \subseteq B$	if and only if	$x \in A$	then	$x \in B$

Introduction to “Formal” Logic

Where is $A \cap B$ now?

Introduction to “Formal” Logic

Let’s prove $A \subseteq A \cup B$

Why so pedantic?

$\{y \mid y \subseteq x\}$

The set of all subsets of x
aka the powerset of x

$\{y \mid y \in x\}$

The set of all elements of x

Aka just x

Why so pedantic?

$x \in x$

x is an element of x
(The set that contains itself)

Does it make sense to write?

Is it ever true?

$\{x | x \in x\}$

The set of all sets that contain themselves

Aka the empty set

Why so pedantic?

$$\{x|x \notin x\}$$

$$y = \{x|x \notin x\}$$

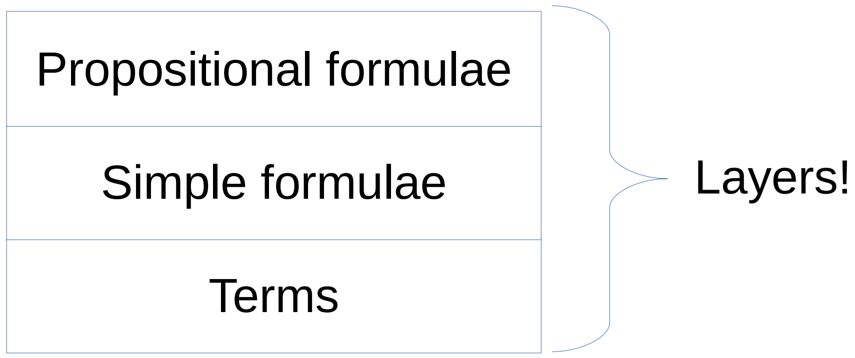
$$y \in y \Rightarrow y \notin y$$

$$y \notin y \Rightarrow y \in y$$

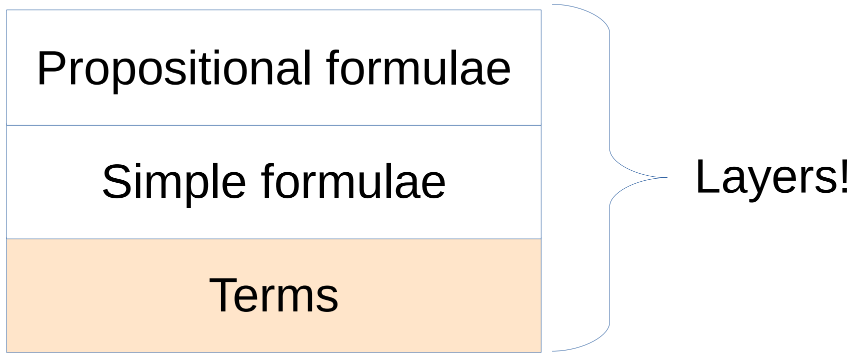
"There is just one point where I have encountered a difficulty"
- Bertrand Russell

Q: Why does this matter?

The language of logic



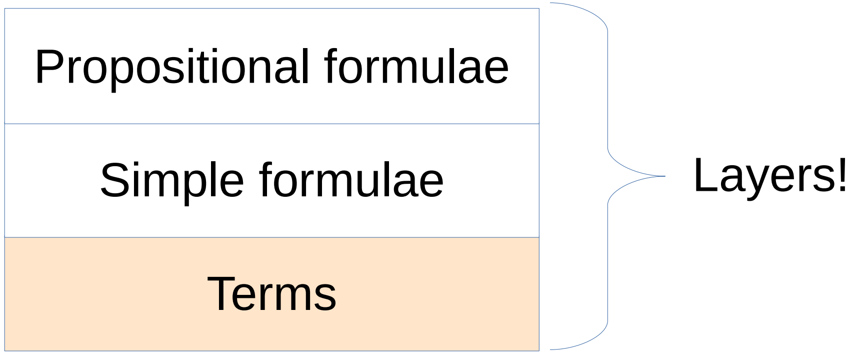
The language of logic



A *term* is either
(a) a variable, or
(b) a constant symbol, or
(c) a function symbol applied to the correct number of other terms.

A function's number of arguments is its *arity*.

The language of logic



A *term* is either
(a) a variable, or
(b) a constant symbol, or
(c) a function symbol applied to the correct number of other terms.

A function's number of arguments is its *arity*.

variables	x, y, ...
Constants	1,{},pi
functions	union, intersection +,-,!

The language of logic

Propositional formulae
Simple formulae
Terms

Layers!

$x+1$

$x \quad 0$

$\sin(x/2)$

Terms have *values*

$x\ y$

$x+$

$x++$

Not terms

The language of logic

Propositional formulae
Simple formulae
Terms

Layers!

t,u are terms

A *simple formula* is a predicate symbol applied to the correct number of (term) arguments.

$$t < u$$
$$t = u$$
$$t \geq u$$
$$even(t)$$
$$t \in u$$
$$false$$

These can be True or False

The language of logic

Propositional formulae
Simple formulae
Terms

Layers!

$\pi \in \mathbb{R}$

$false$

$x \in y$

$1 \leq a/2$

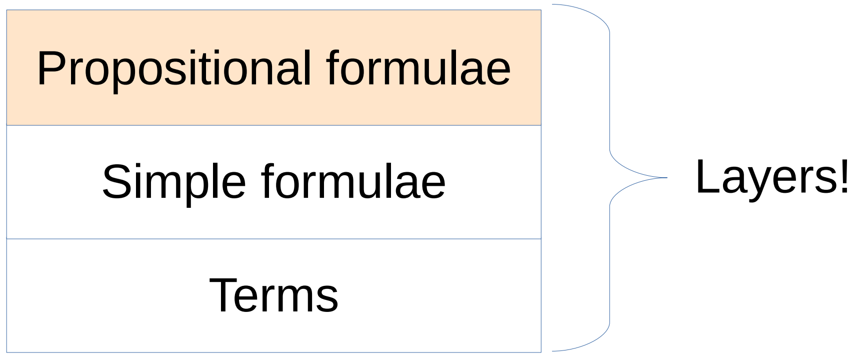
Simple formulae

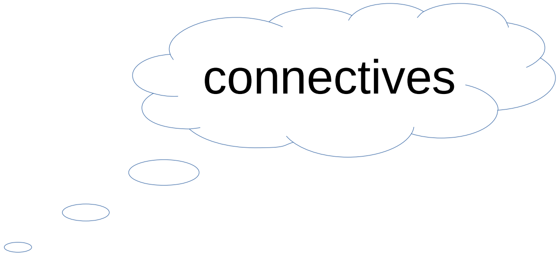
$x \in (y \in z)$

$x + 5$

Not formulae

The language of logic





A *propositional formula* is either
(a) a simple formula
(b) a *propositional connective* applied to
the right number of arguments.

$\wedge$	Conjunction (and)
$\vee$	Disjunction (or)
$\neg$	negation
$\rightarrow$	implication
$\leftrightarrow$	biimplication

The language of logic

Propositional formulae
Simple formulae
Terms

Layers!

Q: Is this formula True?

$$false$$

$$x \in y \land x \geq 2$$

$$n! = n \leftrightarrow (n = 1 \lor n = 3)$$

Truth tables

A	B	A ∧ B
True	True	True
True	False	False
False	False	False
False	True	False

A	B	A → B
True	True	True
True	False	False
False	False	True
False	True	True

A	B	A ∨ B
True	True	True
True	False	True
False	False	False
False	True	True

A	B	A ↔ B
True	True	True
True	False	False
False	False	True
False	True	False

The language of logic: summary

Propositional formulae	Propositional connectives
Simple formulae	Predicate symbols
Terms	Constants, functions, variables

Truth tables define what the propositional connectives mean.

Q: Did the layered, systematic approach help against Russell's paradox?

Calculating with logic

Now we're here →

Calculating with logic

Propositional formulae	Are like the conditions in if-then-else, while
Terms	are like the RHS of assignment statements

We can *calculate* with logic as a thinking tool for programming,

...just as we can use mathematical calculation as a thinking tool for physics.

Calculating with logic

Python syntax.

```
if l <= m < h:
    …
else:
    … #what’s true here?
```

(let’s calculate)

Calculating with logic

```
if l <= m < h:
    thing1
elif m < l:
    thing2
else:
    thing3
```

```
if l <= m < h:
    thing1
elif m >= h:
    thing3
else:
    thing2
```

Are these programs the same?

(let’s calculate)