CS745/ECE-725: Computer-Aided Verification

Overview, Motivation, and Outline

http://www.student.cs.uwaterloo.ca/~cs745
uw.cs.cs745
cs745@student.cs.uwaterloo.ca
Nancy Day

Today's Agenda

- What is this course about?
- Motivation
- Course Organization

What are "formal methods" (FM)?

Also know as "computer-aided verification".

In a nutshell, formal methods strive to provide for computer-based systems (digital-hardware / software) what engineering mathematics has provided in other fields of engineering:

- the ability to express specifications precisely,
- the ability to clearly define when an implementation meets the specification ("correctness"), and
- the ability to understand the specification and the implementation better through "calculations".

Definition adapted from: course notes, G. Gopalakrishnan.

What is computer-aided verification?

In CAV,

- we research techniques for discovering bugs in systems. These techniques can prove that a system has certain desirable functional properties such as: lack of bugs, safety and liveness properties.
- we apply these techniques to all kinds of systems (e.g., software, hardware, protocols, web-based distributed applications).
- we use logical reasoning to symbolically analyze systems without running the system.
- we add these techniques to computer-based tools (CASE and CAD tools), which provide analysis capabilities to users.

CAV provides an analytic side to computer system development.

What is logical reasoning?

According to Webster's (New World, 1984), logic is "the science of correct reasoning".

According to the Free On-Line Dictionary of Computing, "logic is concerned with what is true and how we can know whether something is true".

Using logic, we can reason about a system.

> *Reasoning about situations means constructing arguments about them: we want to do this formally, so that the arguments are valid and can be defended rigorously, or executed on a machine.*
> *– Huth and Ryan*

Trivial Example of Logical Reasoning

Example:

- *If the train arrives late and there are no taxis at the station, then John is late for his meeting.*
- *John is not late for his meeting.*
- *The train did arrive late.*

Were there taxis at the station or not? Therefore, there were taxis at the station.
– Huth and Ryan

This argument has a structure of facts and a conclusion. The conclusion logically follows from the facts.

Are Taxis at the Station?

A1 If train late and no taxis, then John late.

A2 John is not late.

A3 Train is late.

train late	taxis at station	John late	A1	A2	A3
F	F	F			
F	F	T			
F	T	F			
F	T	T			
T	F	F			
T	F	T			
T	T	F			
T	T	T			

Alternative scenario

A1 If train late and no taxis, then John late.

A2 John is late.

A3 Train is not late.

train late	taxis at station	John late	A1	A2	A3
F	F	F	T	F	T
F	F	T	T	T	T
F	T	F	T	F	T
F	T	T	T	T	T
T	F	F	F	F	F
T	F	T	T	T	F
T	T	F	T	F	F
T	T	T	T	T	F

Logical Arguments

> *Example 1: If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.*
> *– Huth and Ryan*

Here's another example of a logical argument:

> *Example 2: If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.*
> *– Huth and Ryan*

These two arguments have the same structure, but use different sentence fragments.

Logical Arguments

> *Example 1: If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.*
> *Example 2: If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.*

Example 1	Example 2
the train is late	it is raining
there are taxis at the station	Jane has her umbrella with her
John is late for his meeting	Jane gets wet

Logical Arguments

The argument can be stated independently of Jane and John and umbrellas and trains and taxis by substituting letters for the sentence fragments as in:

Letter	Example 1	Example 2
p	the train is late	it is raining
q	there are taxis at the station	Jane has her umbrella with her
r	John is late for his meeting	Jane gets wet

The valid argument is then:

if p and not q then r.
not r.
p.

---

q

where the horizontal bar means therefore. This is an argument in propositional logic.

Logical Arguments

Logic is concerned with the structure of the argument, not the meaning of the sentences.

- If 37 is a prime number and I am not the most intelligent person in the world, then 7 is divisible by 3.
- 7 is not divisible by 3.
- 37 is prime

if p and not q then r.
not r.
p.

---

q

Therefore, I am the most intelligent person in the world.

Logic

In a logic, there are rules about which structures of arguments are valid and which are not. In logics, we use symbols to represent the sentences fragments.

A logical argument holds for all values (true or false) of the symbols used in the argument.

Computers are very good symbol manipulators. We can use computers to help us do logical reasoning.

Formalization

Everyone loves my baby, ... but my baby loves only me.

1. *Everyone loves my baby*
   For all people, that person loves my baby.
2. *My baby loves only me*
   For all people, if my baby loves that person, then that person is me.

- From 1, using "my baby": My baby loves my baby.
- From 2, using "my baby": If my baby loves my baby, then my baby is me.
- From modus ponens on 2 using 1: My baby is me!

Source: M. Jackson and D. Gries

Formalization

- First example:
  All Canadians wear toques.
  Counterexample: Joe is Canadian. Joe does not wear a toque.
- Second example:
  There is a person, such that if that person is Canadian, then they wear a toque.
  Joe is not Canadian. Joe does not wear a toque.
  Is this a counterexample?
- *There is a person, such that the person is Canadian and they wear a toque.*
  To prove, must find a toque-wearing Canadian.

Formalization

How would you formalize the sentence: *"I have my umbrella with me unless it is sunny."*

Question: If it is sunny, do I have my umbrella?

Logical Reasoning and Verification?

Using logical reasoning, we can calculate/reason about computer-based systems to determine whether the system has certain desirable properties.

In formal verification, we are checking these properties for all possible behaviours of our model of the system. Thus, formal verification is able to do an exhaustive analysis of the model of the system, whereas testing (or simulation) is only able to check a finite number of behaviours. By reasoning symbolically, exhaustive analysis becomes possible.

Note: in formal verification we are checking a model of the system, not necessarily the real system. For example, formal verification doesn't help us find physical defects resulting from the manufacturing of a chip.

What is computer-aided verification?

Recall:

Formal methods strive to provide for computer-based systems (digital-hardware / software) what engineering mathematics has provided in other fields of engineering:

- the ability to express specifications precisely,
- the ability to clearly define when an implementation meets the specification ("correctness"), and
- the ability to understand the specification and the implementation better through "calculations".

Definition adapted from: course notes, G. Gopalakrishnan.

Logical Reasoning and Verification

Verification involves checking a satisfaction relation, usually of the form of a sequent:

$$\mathcal{M} \models \phi$$

where

- $\mathcal{M}$ is a model (or implementation) describing the possible behaviours of the system
- ϕ is a property (or specification)
- $\models$ is a relationship that should hold between $\mathcal{M}$ and ϕ – what does it mean for the system to be "correct"?

Logic can be used to express the model, property, and relation, and valid arguments of the logic are used to deduce whether the relation holds for the particular model and particular property.

Formal Verification

Verification is:

- formal: the correctness claim is a precise mathematical statement – the model and properties are unambiguous
- definitive: verification either proves or disproves the correctness claim; helps identify the errors in the system

as opposed to:

- testing the actual system on selected inputs
- simulating a model of the system on selected inputs
- manually inspecting the system or model of the system

Computer-Aided Verification

Computer-based tools assist us in this task by:

- automating many of the tedious parts of the proof (e.g., decision procedures),
- doing many of the bookkeeping tasks such as keeping track of what still needs to be proved, and
- checking our proof steps.

Mechanized proofs are those that have been carried out in a computer-based tool.

Levels of Formalization

There are various degrees of using formal methods. In order of increased formality:

1. Using formal notations (those with a precise semantics).
2. Using mechanized support for formal notations, such as syntax checkers, and typecheckers.
3. Using formal proofs usually with mechanization. This mechanization may be fully automated.

There are also semi-formal notations.

Formal methods can be applied to only part of a system.

Analysis vs Synthesis

- The term "Formal methods" usually refers to the analysis side of the analysis/synthesis process of building a system, i.e., checking properties such as safety and correctness.
- But FM can also help with the synthesis side. For example, program analysis techniques are used to justify compiler optimizations.

Opinion Slide

This research area is an elegant mix of theory and practice.

The use of formal methods is a key component in the evolution of computer- and software-engineering as a collection of best practices that result in predictable, reliable computer-based systems.

Today's Agenda

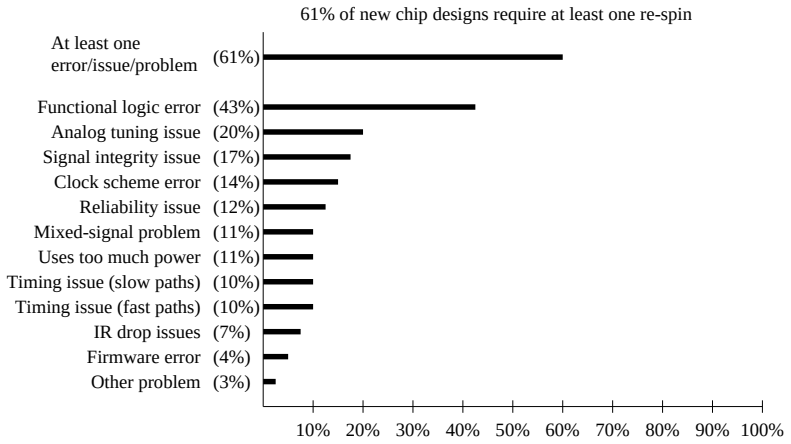
- What is this course about?
- Motivation
- Course Organization

Motivation

It is widely agreed that the main obstacle to "help computers help us more" and relegate to these helpful partners even more complex and sensitive tasks is not inadequate speed and unsatisfactory raw computing power in the existing machines, but our limited ability to design and implement complex systems with sufficiently high degree of confidence in their correctness under all circumstances.
– Amir Pnueli, Turing Award Winner, in forward to Clarke, Grumberg, and Peled, 1999

It takes more effort to verify that digital system designs are correct than it does to design them, and as systems get more complex the proportion of cost spent on verification is increasing (one estimate is that verification complexity rises as the square of design complexity).
- Michael J. C. Gordon, in

Motivation



Source: Aart de Geus, Chairman and CEO of Synopsys. Keynote address. Synopsys Users' Group Meeting, Sep 9 2003, Boston USA.

Problems

The development of computer-based systems:

- often run behind schedule or get cancelled
- may result in products that don't do what you expect them to do
- often have subtle bugs. When used in safety-critical applications, these bugs can cost lives or huge amounts of money.

No Guarantees

Because of these problems, most commercial software comes with no guarantees.

Two good references on the challenges of using computers in safety or mission-critical applications:

- Risks Forum moderated by Peter G. Neumann and his book "Computer-Related Risks", Addison-Wesley, 1995. `http://catless.ncl.ac.uk/Risks/`
- "Fatal Defect: Chasing Killer Computer Bugs" by Ivars Peterson, Vintage Books, 1996.

Loss of Life: Therac-25

A computer-controlled radiation therapy machine called the Therac-25 made by Atomic Energy of Canada overdosed six people between June, 1985 and January 1987.

In some cases the only indication that something was wrong was the cryptic message:

malfunction 54

The error was a timing problem on data entry.

Loss of Life: Therac-25 (cont'd)

The Therac-25 could deliver radiation as either a beam of electrons or a beam of X-rays. If the operator entered "x" for x rays, the setting of the magnets took 8 seconds. If the operator discovered she had made a mistake and fixed the entry to be "e" within that 8 seconds, even though the screen reflected the change, the change did not affect a part of the program.

See: Leveson and Turner, "An investigation of the Therac-25 accidents", 1993.

FAA Advanced Automation System

> *FAA's major modernization project, the Advanced Automation System (AAS), was originally estimated to cost $2.5 billion with a completion date of 1996. The program, however, experienced numerous delays and cost overruns, which were blamed on both FAA and the primary contractor, IBM. In 1994, FAA cancelled part of the program and split the remaining systems into three phases, and in several cases, re-bid the contracts. . . . According to the General Accounting Office (GAO), almost $1.5 billion of the $2.6 billion spent on AAS was completely wasted.*

Originally from:
http://www.house.gov/transportation/Press2001/release15.html
(link no longer works)

Pentium FDIV bug

The floating point processor of Intel Pentium chips manufactured before a certain date had a bug in the FDIV operation. Approximately 2 million chips had the flaw. The FDIV operation didn't always give precise results. According to sources on the web, the bug could be seen by entering the following formula in the Windows calculator:

$$(4195835/3145727) * 3145727 - 4195835)$$

It produced 512 instead of 0.

Pentium FDIV bug (con'd)

Intel identified the problem in testing, summer, 1994.

Prof. Thomas Nicely, Lynchburg College, first identified in email to colleagues Oct, 1994.

EE Times had an article on it in Nov, 1994.

> *"This was a very rare condition that happened once every 9 to 10 billion operand pairs," said Steve Smith, a Pentium engineering manager at Intel. Intel's Smith emphasized that the anomaly would not affect the average user. Speaking of Nicely, Smith said: "He's the most extreme user. He spends round-the-clock time calculating reciprocals. What he observed after running this for months is an instance where we have eight decimal points correct, and the ninth not showing up correctly. So you get an error in the ninth decimal digit to the right of the mantissa. I think even if you're an engineer, you're not going to see this."*
> *– EE Times, November 7, 1994, Issue 822, page 1*

Pentium FDIV bug (con'd)

Intel offers to replace it for those customers based on "need".

Stock prices fell, great concern.

Dec, 1994, Intel offers to exchange the processor for anyone who asks.

Cost to Intel estimated at $475 million.

Intel hired many people who did formal verification after this incident and they continue to carry out research in hardware verification and are applying these techniques in practice.

See: The Pentium FDIV bug - a Picture

`http://kuhttp.cc.ukans.edu/cwis/units/IPPBR/pentium_fdiv/pentgrph.html`

The Pentium Papers

`http://web.ccr.jussieu.fr/ccr/Documentation/Calcul/matlab5v11/docs/00000/0002d.htm`

The Ghost of the Pentium FDIV Bug

From The Risks Digest, Vol. 19, Issue 4, 4 April 1997
(`http://catless.ncl.ac.uk/Risks/19.04.html#subj3`)

> *The ghost of the Pentium FDIV bug*
> *Frank Solomon Fri, 04 Apr 1997 09:09:41 -0500*
> *It seems that the ghost of the FDIV bug lives on in Excel spreadsheets created using Pentiums affected by the problem.*
> *I finally got rid of my Intel Pentium computer with the FDIV bug. Last night, while checking out my new Pentium Pro computer with Microsoft Excel 97 I decided to open up the spreadsheet which demonstrates the Pentium Floating Point divide bug. I was surprised to find that that the calculation of:*
> *4195835 - (4195835/3145727)*3145727*
> *within the spreadsheet still showed the answer 512 instead of zero.*

The Ghost of the Pentium FDIV Bug

I pressed the recalculate key (F9) to no avail. So then, I retyped the formulas in neighboring cells. Where I had retyped the formulas the answer(s) were correct, but the same formulas that had been saved from when I had done the calculation on the defective Pentium still showed the wrong answer! In other words, here, on the same spreadsheet was the same problem with two different answers, one correct and one incorrect.
The only way I could find to "correct" the incorrect answers was to retype the formulas over the originals.
It seems to me that this "ghost" could represent some risk since it is logical to assume that if you're no longer using a defective Pentium, you needn't be concerned about wrong answers on the spreadsheets you've moved to a new machine. This obviously is not so.
I've sent in a bug report to Microsoft as of this morning.
Frank Solomon

Banking

... February 1994, automated teller machines (ATMs) at Chemical Bank in New York City mistakenly deducted a total of approximately *$15 million dollars* *from about a* *hundred thousand customer accounts*. *Until the problem was discovered, any customers making withdrawal were charged double the withdrawal's actual amount on their accounts, although the printed transaction slip showed the correct amount. Only those people who later check their balance – and knew what it should have been – realized there had been an error. The culprit proved to be a flawed instruction –* *a single line* *in updated computer program the company had installed day before the problem surfaced.*
– Peterson, 1996.

Windows XP

Microsoft released Windows XP on Oct. 25, 2001. That same day, in what may be a record, the company posted 18 megabytes of patches on its Web site: bug fixes, compatibility updates, and enhancements. Two patches fixed important security holes. Or rather, one of them did; the other patch didn't work. Microsoft advised (and still advises) users to back up critical files before installing the patches. Buyers of the home version of Windows XP, however, discovered that the system provided no way to restore these backup files if things went awry. As Microsoft's online Knowledge Base blandly explained, the special backup floppy disks created by Windows XP Home "do not work with Windows XP Home".

Originally from:
http://msnbc.com/news/768401.asp?cp1=1#BODY (link no longer works)

Do you trust your computer system?

The real wonder is that the system works as well as it does.
(Peterson, 1996)

Consider the future applications of computers:

- free flight
- smart homes
- patient monitoring
- automated highways
- etc.

Smart Fridges

- Watch television (remote control)
- Play music, download MP3s
- E-mail or web surfing
- Built-in digital camera
- Leave messages in video or audio
- Keep track of birthdays
- Organize recipes
- Keep track of stored foods and monitor expiration dates
- Self-diagnostics

Originally from:
http://www.lgappliances.com/product.asp?category_id=1 (link no longer works)

Security: SmartCards

Smart cards, the size of a credit card, have a microprocessor and memory, along with a mini operating system. They can run multiple applications, which may be downloaded after the card is in use. (This feature is currently disabled.) These "applets" can carry out various functions such as being an "electronic wallet", carry health information, etc.

Because of the high security considerations, a European project is attempting to verify the code and operating system of these cards for "non-interference" between applications. For these companies "security is their product".

See:
Verificard: `http://www.verificard.org`

Why study CAV?

CAV is an area of research whose goal is to create techniques for analyzing systems to find subtle, critical logic and safety errors. As computer-based systems:

- become more complex, and
- are used in mission-critical applications (e.g., space shuttle, radiation treatment, stock market),
- the effort and money spent on testing and maintenance takes up more of the system development cycle,
- many bugs can be traced to faults in the requirements/models,

these techniques are necessary to ensure the development of reliable and safe systems.

These techniques are gradually being incorporated into CASE and CAD tools.

Companies using Formal Methods

- Intel
- Siemens
- BT
- AT&T
- Lucent
- Nvidia
- Nortel
- Airbus
- IBM
- NASA
- Motorola
- Cadence
- Synopsis
- etc.

See:

- Craigen, Gerhart, and Ralston, "Formal methods reality check: Industrial Usage"
- Intel Strategic CAD Labs:
  http://www.intel.com/research/labs.htm#strategiccadlabs

Caveats

The use of formal methods does NOT solve all problems.

Formal methods form a part of a methodology that involves:

- "best practice" system development
- consideration of safety from the beginning,
- adequate testing and inspection processes,
- appropriate assessment of the context of the system and,
- the suitability of human interfaces and practises.

Also note that formal methods tools and users can make mistakes.

Today's Agenda

- What is this course about?
- Motivation
- Course Organization

Class Time and Office Hours

- Class Time: 3 hours/wk
  Tue/Thu 9-10:30 MC 2034
- Office Hours: TBA

Course Organization

This course will cover an introduction to computer-aided verification.

- logics,
- logical representations of systems, and
- computer-based support for reasoning in these logics.

We will work from:

- from highly expressive logics with general, non-automatic proof procedures to special cases where the logics are less general but the proof procedures are more automated and efficient

Course Goals

- To introduce you to the fundamental concepts of formal methods
- To introduce you to the range of topics within formal methods and to provide a map to guide you toward applications and future research in this area.
- To give you hands-on experience with formal methods tools. FM is an area that combines both theory and practise.
- To learn what research is and how to do it.

Course Goals

To achieve these goals we will:

- cover fundamental topics in detail
- talk a little bit about many other topics
- demonstrate and use formal methods tools (assignments)

Course Outline

Here is an approximate outline of topics that we will cover in this class. This list is probably ambitious. We have about 36 hours of class meetings.

- Outline, Motivation, Overview (today)
- Foundations (review): propositional and predicate logic
- Major Topics:
  - Interactive Theorem Proving (higher order logic)
  - Temporal Logic and Model Checking (automatic)
- Applications: (software, hardware, etc.)
- Advanced Topics: (decision procedures, additional model-checkers, verification strategies)

Course Outline

- Possible advanced topics
  - Decision Procedures for Propositional Logic (SAT, Binary Decision Diagrams, etc.)
  - Decision Procedures for subsets of first order logic
  - Symbolic Trajectory Evaluation
  - Decomposition Strategies, Abstraction, and Symmetry Reductions
  - Rewriting
  - Combining Decision Procedures and Theorem Proving

Course Outline

In addition there will be:

- Paper Presentations (30min each)
  - Choose a paper from the literature (a list is available on the course web page) and present and critique the material in the paper.
    - Papers can be chosen on a first-come, first serve basis.
    - Presentations will be distributed throughout the term.
  - Explain the material in the paper in the context of what we have learned in class.
  - Everyone evaluates everyone else (which contributes to the grade assigned the presentation and also contributes to your participation grade).
    - 65% by the instructor
    - 35% by your classmates
  - The evaluation form is on the course web page.

Evaluation (Preliminary)

Assignment 1 (theorem proving – HOL)	15%
Assignment 2 (model checking – SMV)	15%
Assignment 3 (automated first-order proof – Alloy)	15%
Paper critique (in-class)	5%
Paper presentation	20%
Paper report	20%
Participation	10%

There will not be a midterm or a final exam.

Participation

10% of your final grade will be based on participation. This involves:

- Attendance (which includes not being late to class)
- Attention
- Asking questions
- Helpful comments on the presentation evaluation forms.

Expected Background

Subjects that you should have some familiarity with:

- propositional, predicate logic
- sets, functions, relations
- induction
- computability theory
- recursion
- order theory: partial order, equivalence classes, lattices
- automata theory

Web Page and Newsgroup

Lecture notes will be on-line at:
http://www.student.cs.uwaterloo.ca/~cs745

This web site will also have links to tools, references, and other course information.

Newsgroup: uw.cs.cs745

Sources

Course material will be drawn from research articles and the texts. Citations in the notes will point you towards these references. Here are some good references, but you do not need to buy these texts.

- Edmund Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999.
- Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1996. Second Edition.
- John Kelly, *The Essence of Logic*. Prentice Hall, 1997.

Sources

- Michael Huth and Mark Ryan. *Logic in Computer Science* Cambridge University Press, 2004.
- NASA. *Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems Volume II: A Practitioner's Companion.* May 1997. http://eis.jpl.nasa.gov/quality/Formal_Methods/

Sources

- Thomas Kropf, *Introduction to Formal Hardware Verification*. Springer, 1999.
- Carl-Johan Seger, An Introduction to Formal Hardware Verification, UBC CS TR 92-13,June 1992. http://www.student.cs.uwaterloo.ca/~cs745/Ref
- Doron Peled, *Software Reliability Methods*, Springer, 2001.
- Ganesh Gopalakrishnan, *Computation Engineering: Applied Automata Theory and Logic*, Springer, 2006.

Sources

- Klaus Schneider, *Verification of Reactive Systems*. Springer, 2004.
- B. Berard et al., *Systems and Software Verification*. Springer, 2001.

Equipment

There are several servers available for use in the assignments and projects (all are Intel chips running Debian Linux).

- mudge.cs.uwaterloo.ca (4G RAM, 2CPUs, 1.8 GHz)
- tumbo.cs.uwaterloo.ca (8G RAM, 2 CPUs, 1.4 GHz)
- quadra.cs.uwaterloo.ca (8G RAM, 4 CPUs, 1.4 GHz)
- gooch.uwaterloo.ca (4G RAM, 2 CPUs, 1 GHz)

The software used in the assignments will be set up here. Software used in your projects can also be set-up here. The servers are available for remote log-ins.

All of the tools used in this course are academic prototypes. Most can be downloaded, installed and run on your own machine.

Formal Methods at Waterloo

If you are interested in computer-aided verification for possible future research, you should consider also attending the Waterloo Formal Methods research group meetings.

Information on these meetings can be found at

http://www.watform.uwaterloo.ca

Information on the activities of the research group can be found there also.

Information on joining the WatForm seminar mailing list is also available at this website.

Reminder: Ask Questions

ASK QUESTIONS!

Research is about asking questions.
Ask lots of questions.
Ask questions when you don't understand a term.
Ask questions when you don't understand an algorithm or the reasoning leading to a conclusion.
Ask questions when you see a connection between different parts of the course.
Ask questions about anything.

(Of course, you may be asked to answer them yourselves!)

Also, suggestions for how to improve the course are always welcome. Please point out errors in the notes.