CS Qualifier Description

September 22, 2008

Abstract

UIC CS Faculty will administer a qualifier exam starting Spring 2009. The qual exam is divided into four areas. Students demonstrate competency in a minimum of three areas. Some commonly agreed-on terms during the discussions on the qualifier are also recorded in this document. More logistics and details will be added by the graduate committee to this document.
1 Exam details

- Figure 1 summarizes the proposal.

- The qual is into four areas, theory & foundations, software systems & languages, core systems & networking, and Artificial Intelligence & Human Centered Computing.

- Each area will administer an exam. Faculty from each area will decide on the nature of questions in each exam. Each exam will be for at most 2 hours. The syllabus for each area is attached in the appendix.

- Students choose (at least) three areas of the four to demonstrate competency, subject to the following constraints:
  - The theory & foundations area is mandatory.
  - The student demonstrates either:
    * three “full” passes (full and conditional passes are explained in Section 1.1).
    * two “full” passes plus one “conditional pass”.
  - Each area exam can be taken at most twice.

- Exams can be taken in different semesters with constraints that each student must take at least one exam by the second semester. For example, a student coming in during Fall 2008 must take at least one area exam offered in Spring 2009.

- The first exam will be offered in Spring 2009.

1.1 Exam Results

Each exam has one of the following three results for a student.

1. Pass (a “full pass” demonstrating clear competency).

2. Conditional Pass (demonstrates some competency that can be improved by coursework).

3. Fail (demonstrates low competency).

Each area will set its own cut-offs for conditional pass and fail.

The idea is that the conditional pass can be given to students who didn’t do well enough to demonstrate clear competency in an area, and can supplement their knowledge through coursework and demonstrate competency through grades.

This way, each student must aim to be in the “full pass” or at least be in the “conditional pass” range for the exam, and not fail.
Conditional pass.

- If student gets a conditional pass for any area, then the qual committee will examine the student’s performance in various topics in that area exam, and will suggest a 400 level course for the student. If the student has already taken a suggested class, then faculty may suggest another 400 or 500 level class.

- The student must take the class and get an A in the course to demonstrate competency.

- Must finish any conditional coursework before the end of the 5th semester.

2 Faculty Memorandum of Understanding

The following notes summarize the understanding of the faculty at the time of writing this document. These terms were agreed on by faculty during the May 12 meeting as well as over email discussions.

- It is recommended that questions be of short answer type (or questions on the same topic split into several short answer questions). Questions with essay type answers are discouraged.
• For the theory exam, questions will focus mainly on problem solving, and will not be “theorems & proofs” oriented.

• In general, the exam will be administered at the beginning of the semester, 2-3 weeks after the start of the semester.
Proposed Theory Qual Syllabus

Foundations

• Set Theory

• Basic counting, combinatorics and discrete probability

• Proof Techniques
  – Mathematical Induction: Strong, weak, structural
  – Proof by contradiction

• Propositional Calculus

• Predicate Logic

Data Structures and Algorithms

• Linear Data Structures: lists, stacks, queues.

• Analysis of Algorithms
  – Elementary operations
  – Big-Oh, Big-Omega, Big-Theta
  – Best, worst and average case analysis
  – Determining and analyzing first order recurrence relations

• Trees and Priority Queues
  – Binary Search Trees
  – Balanced Binary Search Trees
  – Tree traversals
  – Union-Find Data Structures
  – Binary Heaps

• Sorting, Searching and Selection
  – Binary search
  – QuickSort
  – QuickSelect
  – MergeSort
  – Lower-Bound on Comparison Based Sorting

• Graphs
  – Data Structures - adjacency matrices, adjacency lists
  – Depth First Search and Applications
– Topological Sort and Applications
– Breadth First Search
– Single-Source Shortest Paths
  * Dijkstra’s Algorithm
  * Bellman-Ford Algorithm
– All-pairs Shortest Paths
– Minimum Spanning Trees

• Dynamic Programming

• Greedy Algorithms
  – Huffman codes
  – Minimum Spanning Trees

• Divide and Conquer Algorithms
  – QuickSort
  – MergeSort
  – Selection

Theory of Computation

• Models of Computation

• Finite State Automata (FSA) and Regular Expressions
  – Deterministic FSAs
  – Nondeterministic FSAs
  – Regular Expressions
  – Equivalences
  – **Note:** The Pumping Lemma for Regular Languages will *not* be tested.

• Context-Free Grammars

• Undecidability

• Reductions and NP-completeness
  – Polynomial Time Reducibility
  – Complexity Classes P, NP, NP-complete
  – Certificates and Verification
Software Systems & Languages Exam Syllabus

Students must choose 3 of the 4 areas below. If a student answers all four areas, the area corresponding to the lowest grade will be discarded.

1. Compiler Design
   - Design and Implementation aspects of Compilers
   - Formal description of programming languages
   - Lexical analysis
   - Syntax analysis
   - Symbol tables
   - Abstract syntax trees
   - Syntax-directed translation and attribute grammars
   - Semantic analyses (type checking)
   - Memory allocation
   - Code generation
   - Interpreters

2. Database Systems
   - DBMS Principles
   - Database Conceptual Design
   - Relational Model
   - SQL
   - Relational Database Design
   - Indexing and Hashing
   - Query Processing and Optimization
   - Concurrency Control and Recovery

3. Programming Languages
   - Important programming paradigms such functional, object-oriented and logic programming paradigms
   - Important design and implementation considerations behind LISP, ALGOL, ML, Prolog, and C++
   - Programming language concepts:
     - Type systems: Static vs. dynamic typing
     - Scope rules: Static vs. dynamic scope
     - Overloading
     - Parameter Passing: call by value, reference and name
     - Object allocation: Static, automatic, and dynamic allocations
     - Object deallocation: Automatic garbage collection vs. programmer-controlled
     - Run-time environment
• Programming language constructs:
  – Literals
  – Identifiers
  – Expressions
  – Control Structures
  – Program units (function, procedures, methods)
  – Recursion
  – Classes and inheritance: instances, messages, methods, polymorphism, abstract classes, multiple inheritance.

4. Software Engineering

• Software Life cycles
  – Object Oriented Life cycles: Unified Process
  – Classical Lifecycles: Waterfall Model, Spiral Model, Agile Processes and Extreme Programming

• Requirements: Desirable properties of Specifications

• Design: Architectural Design and Detailed Design

• Testing
  – Unit testing: Black-box testing, white-box/glass-box testing, statement coverage, branch coverage, DU-path and path coverage
  – Integration testing: Top-down, bottom-up and sandwich testing
  – System testing: Alpha testing, beta testing, stress testing, regression testing

• Verification
  – Partial correctness assertions
  – Loop invariants
  – Proofs
1 Architecture/Hardware

- gates, flip/flops, combinational and sequential circuits
- arithmetic (ripple and carry lookahead adders, AC:MQ multiplier).
- datapath
- clocking disciplines
- control unit
- pipelining
- memory hierarchy (direct mapped, set-associative, and fully associative caches; main memory, pages/line replacement policies, architectural issues in virtual vs physical address spaces)
- functional units
- multiple issue and VLIW architectures

2 Operating systems

- process
- OS-Arch: privilege instructions, memory management, interrupts
- system calls: fork, exec, wait, read/write,
- interprocess communication
- deadlock and starvation
- synchronization: semaphores and monitors
- scheduling
- virtual memory
- virtual machines
- file storage
- access controls
3 Networks

- sockets programming
- IP
- TCP
- UDP
- DNS
- Email, HTTP / Web
- Ethernet
- routing and forwarding: link-state, distance-vector, ARP
- reliable connections (recovery from packet loss, corruption, duplication)
- congestion control (eg. van jacobsen)
- multicast
Human-Centered Computing Area

Artificial Intelligence
  - Intelligent agent architectures
  - Search techniques
  - Predicate and first-order logics
  - Knowledge representation
  - Machine learning
  - Planning
  - Uncertain knowledge and reasoning
  - Rationality and decision-making

Computer Graphics
  - Homogeneous coordinates, geometric transformations, normalization
  - Scan conversion, polygon filling, clipping
  - Hierarchies, Matrices
  - 3D graphics
  - Human visual system
  - Color models
  - Lighting and shading models
  - Visible surface determination, depth buffer, back-face culling
  - Texture mapping
  - Raytracing
  - Depth cueing and stereoscopic computer graphics

Human Computer Interaction
- Designing the User Interface 4th ed., Schneiderman, Plaisant
- The Design of Everyday Things, Don Norman
- Interaction Design: Beyond Human Interaction, Sharp, Rogers, Preece
  - Affordances
  - Conceptual models
  - Gulfs of execution and evaluation
  - 7 stages of action
  - Constraints
  - Diversity
  - Human ability / physiological issues
  - Golden rules
  - Form factors and interaction techniques
  - Models
  - Interface design
  - Evaluation techniques
  - Collaboration issues