Data Hiding in ML

11-17-2004

Opening Discussion

- What did we talk about last class?
- For assignment #7, submit what you have and start working on assignment #8. I've made it so that you can read my version of the XML parser in the code directory. Just so you know, my code might require some changes for the validating parser.
- Does anyone have any questions about assignment #8? Does everyone understand the concepts of graphs and octrees?

Sharing

- Sharing is a concept that we can use in a signature to assure that two different names refer to the same type. It can be done with general types or structures.
  - sharing type <t1> = <t2> = ... = <tN>
  - sharing <struct1> = <struct2> = ... = <structN>
- This is useful when a signature includes multiple structures that have types in them. If those internal structures are related, we need to be able to specify that their components have the same type.

Information Hiding

- One of the most powerful aspects of modules is that they allow information to be hidden. This information can either be functions or types.
- This gives us power because it relates back to the whole idea of separation of interface and implementation. Anything that is hidden can be changed without worrying about breaking outside code.
- ML provides several ways for modules to hide parts of their implementations.

Hiding with Signatures

- We have already discussed the ability of ML to hide elements of structures by defining signatures that leave out those elements.
- We used this with our BST to hide the deleteMin method so that outside code couldn't accidentally call it with Empty.
- We can specify a signature with this syntax.
  - structure <ident> : <signature> = struct ... end

Abstract Types

- ML also provides abstract types. These are datatypes where the constructors are hidden from the outside world. To make them useful, when you provide the datatype specification, you also provide the functions that will be able to access the constructors.
  - abstype <datatype def> with ... end
- Nothing outside of with and end will have access to the datatype constructors. In fact, ML won't even print the type outside.

Local Definitions

- Yet another way to hide certain elements of code is with local definitions. A local definition makes some definitions only visible to a set of other definitions.
  - local <defs1> in <def2> end
- The definitions in <defs2> can use those in <defs1>, but outside code will only be able to see what is in <defs2>.
- Using this, we could make deleteMin and the exception in our BST local to delete so they can't be see from the outside.

Opaque Signatures

- ML97 also added one other feature that can hide certain parts of structures. If the signature of a structure is given with :> instead of just :, all of the abstract types (things specified with type) that would be in the signature are hidden from outside users.

Minute Essay

- Which form of information hiding in ML to you think is best? Why? What are the strengths of that method?
- Quiz #5 will be next class and remember that assignment #8 is due on Tuesday.