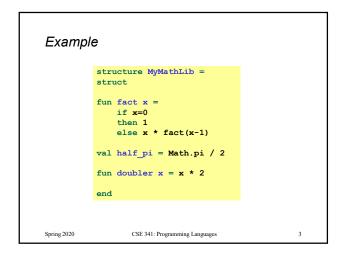
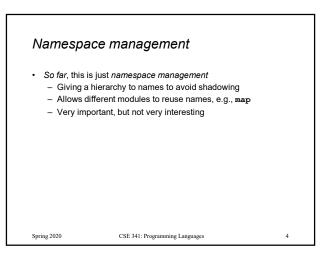
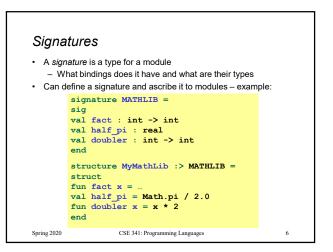
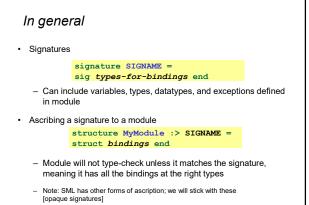
W PAUL G. ALLEN SCHOOL of computer science & engineering Modules For larger programs, one "top-level" sequence of bindings is poor - Especially because a binding can use all earlier (nonshadowed) bindings So ML has structures to define modules CSE341: Programming Languages structure MyModule = struct bindings end Lecture 10 Inside a module, can use earlier bindings as usual ML Modules - Can have any kind of binding (val, datatype, exception, ...) Outside a module, refer to earlier modules' bindings via ModuleName.bindingName Brett Wortzman - Just like List.fold1 and Char.toLower; now you can Spring 2020 define your own modules Spring 2020 CSE 341: Programming Languages 2



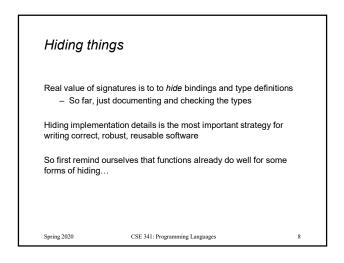


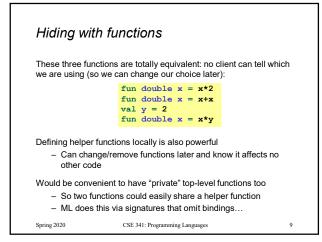
Optional: Open 9. Considering ModuleName to get "direct" access to a consule solutions. 9. Conserver, use a convenience; often bad style. 0. Conserver, user a convenience; often bad style. 0. Director access to access to

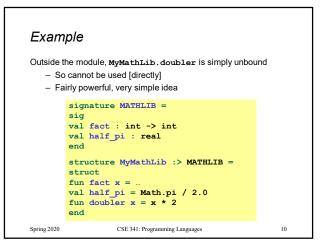


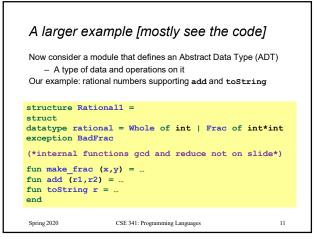


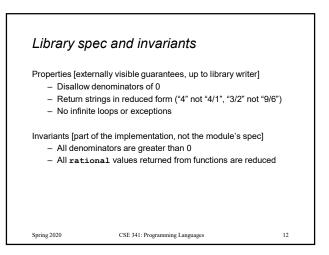


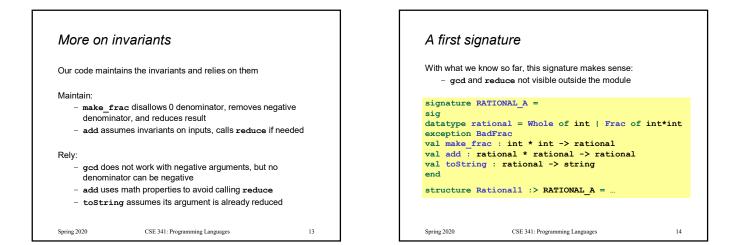


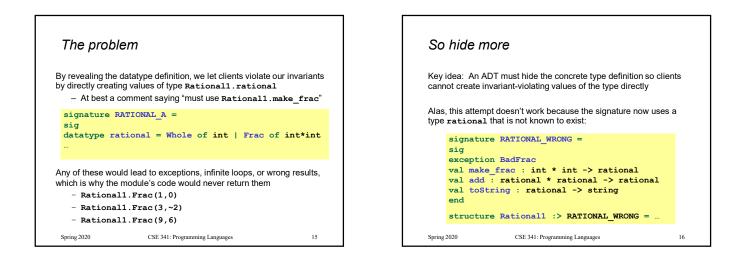


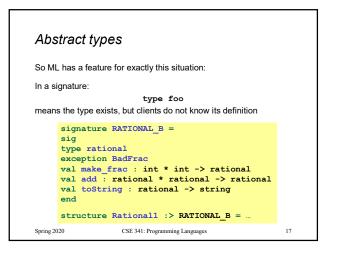


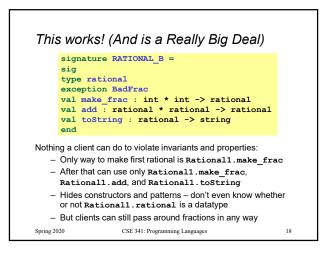


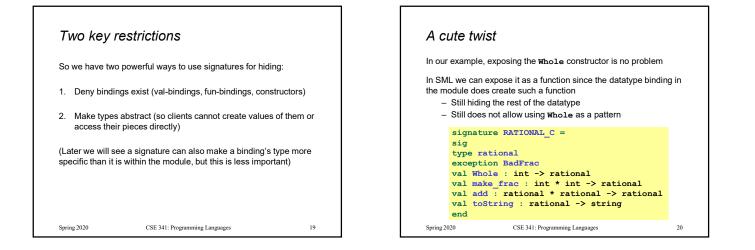


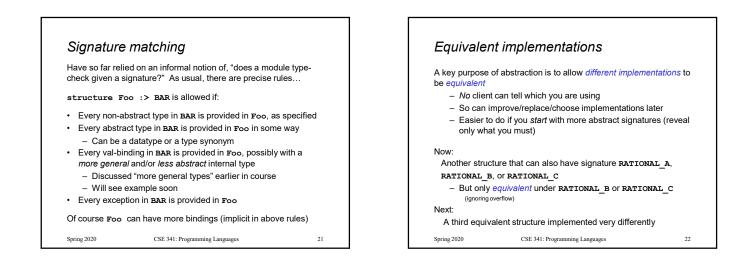






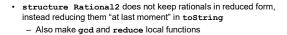






Equivalent implementations

Example (see code file):

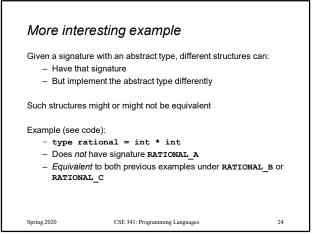


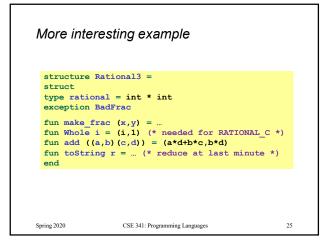
- Not equivalent under RATIONAL_A
 - Rational1.toString(Rational1.Frac(9,6)) = "9/6"
 - Rational2.toString(Rational2.Frac(9,6)) = "3/2"
- Equivalent under **RATIONAL_B** or **RATIONAL_C**
 - Different invariants, but same properties
 - Essential that type rational is abstract

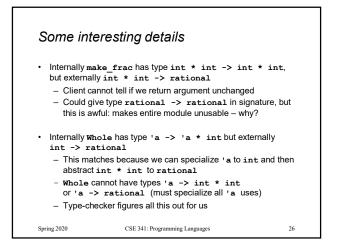
Spring 2020

CSE 341: Programming Languages

23







Can't mix-and-match module bindings Modules with the same signatures still define different types So things like this do not type-check: Mational1.toString(Rational2.make_frac(9,6)) Ational3.toString(Rational2.make_frac(9,6)) Different modules have different internal invariants! Mational1.rational looks like Rational2.rational, but clients and the type-checker do not know that Ational3.rational is int*int not a datatype!