CSE341: Programming Languages

Lecture 24
Racket Modules, Abstraction with Dynamic Types; Racket Contracts

Dan Grossman

Fall 2011

Another modules lecture

- Recall lecture 12: SML modules. Key points:
  - Namespace management for larger programs (structures)
  - Hiding bindings inside the module (**`gcd`**, **`reduce`**)
  - Using an abstract type to enforce invariants

```
signature RATIONAL =
sig
type rational
exception BadFrac
val make_frac : int * int -> rational
val add : rational * rational -> rational
val toString : rational -> string
end

structure Rational :> RATIONAL = …
```

Racket is different

- More flexible *namespace management*
  - Convenient ways to rename during export/import
  - (In other languages, could write wrapper modules)
- Dynamic typing still has ways to create *abstract types*
  - Just need to be able to make a new type at run-time
  - This is what **`struct`** does; Scheme has nothing like it
- By default, each file is a module
  - Not necessary but convenient
- State-of-the-art *contract system*
  - Arbitrary dynamic checks of cross-module calls with blame assignment