

Class 30: Preconditions, Revisited

Held: Monday, 26 October 2009

Summary: We consider preconditions and programming techniques for ensuring that preconditions are met.

Related Pages:

- EBoard.
- Lab: Verifying Preconditions.
- Reading: Verifying Preconditions.

Notes:

- Welcome back from break! I hope you are feeling refreshed.
- Reading for tomorrow: Local Procedure Bindings and Recursion.
- The reading for today accidentally included some notes on an image model (spots) that we have not covered this semester. I apologize. That issue has been addressed.
- The distribution on the last quiz was bimodal, so we'll go over it a bit.

Overview:

- Verifying preconditions.
- The error procedure.
- Husk and Kernel programming.

Preconditions

- Remember: Documentation is a contract, be careful.
- Even if you don't write documentation (or don't write careful documentation), you should reflect on your expectations for parameters.
- We call those expectations the preconditions of the procedure.
- In some cases, you may want to explicitly test to ensure that the preconditions are met.
 - Testing lets you provide more useful error messages.
 - Testing prevents dangerous results from being returned.

Reporting Failure

- When a precondition fails, you want to stop computation immediately and report the error.
- In Scheme, you can do so with `error` or `throw`
 - The one to use depends on the version of Scheme you are using.
 - Our Scheme interpreter uses `error`.

Husk and Kernel Programming

- Particularly for recursive procedures, it is inefficient to check preconditions at every recursive call
 - If the preconditions were met for the first call, they should be met for every subsequent call.
- Hence, programmers tend to use what I refer to as “Iowa’s Great Contribution to Programming”: The Husk-and-Kernel approach
 - The husk checks the preconditions and, if all preconditions are met, calls the kernel.
 - The kernel does the real work.
- Corn serves as the metaphor: The husk protects the kernel, and the kernel is the valuable part.
 - And no, Husk-and-Kernel programming was not invented in Iowa.

Copyright © 2007-9 Janet Davis, Matthew Kluber, Samuel A. Rebelsky, and Jerod Weinman. (Selected materials copyright by John David Stone and Henry Walker and used by permission.) This material is based upon work partially supported by the National Science Foundation under Grant No. CCLI-0633090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.