

The DrScheme Programming Environment

Summary: We examine the program development environment in which we will work for most of the semester, which the course is conducted.

Disclaimer: For this reading and the next, we have a bit of a “chicken and egg problem”. That is, it’s difficult to introduce the environment in which you will write algorithms without first introducing the language in which you will write those algorithms. At the same time, you cannot start learning the language until you’ve learned a bit about the environment. In this reading, and the corresponding lab, we will emphasize the environment, but teach you a bit about the language, too.

Contents:

- Introduction: Program-Development Environments
- Running DrScheme
- An Overview of The DrScheme User Interface
- Detour: A Bit of Scheme
- The Interactions Pane
- Editing in the Interactions Pane
- The Definitions Pane
- Saving and Restoring Definitions
- Concluding Thoughts

Introduction: Program-Development Environments

One of the main activities of many computer scientists is that of *programming*, expressing algorithms in a form understandable to the computer. (Even computer scientists who emphasize other issues often end up needing to build programs to help support arguments or to provide concrete evidence for theories.) Much of this semester, you will be writing programs as a way of learning about important concepts. Programming is also a skill that you can apply in a variety of contexts.

At this point, most programmers develop their code in what is called an *program-development environment*. Program-development environments let you write code, test bits of the code, format your code for easy readability, obtain documentation on built-in procedures, and so on and so forth.

In this class, we will use a program-development environment called *DrScheme*. DrScheme is a leading program-development environment for Scheme. It was developed for teaching, but professional Scheme programmers use it, too.

Running DrScheme

You start DrScheme by clicking on the small blue-and-red lambda (λ) (a bit like an upside-down, lower-case y). that appears in the task bar. If you have no such icon, find a teaching assistant or teacher for help.

The first time you start DrScheme, you will need to configure it. You configure two key aspects: The human language in which DrScheme interacts with the programmer and the version of Scheme that it uses. Different versions of DrScheme seem to behave a bit differently (and the same version sometimes seems to behave a bit differently for different people), so a complete description of what to do is neither possible or necessary. Suffice to say that it will be obvious how you choose the human language and may be a bit less obvious how to select the version of Scheme.

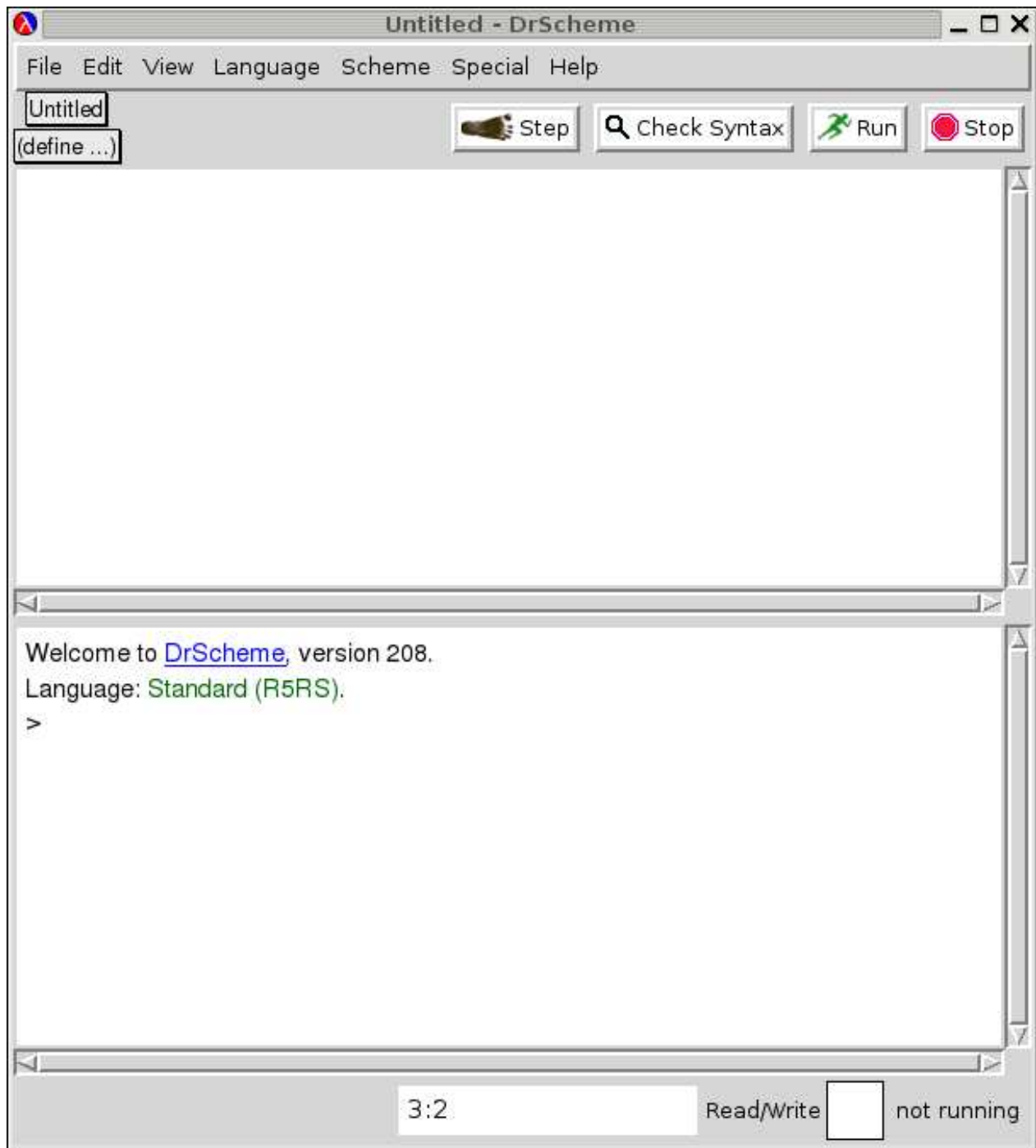
Why do we have to select a version of Scheme? Because as the Scheme Programming Language has evolved, a few dialects have evolved. (You certainly expect that dialects exist for human languages; it turns out to happen for computer languages, too.) The DrScheme environment also has a few versions that include additional “built-in” functions and a few versions that are simplified for novice programmers. In this course, we will use a fairly rich version of Scheme, which the designers of DrScheme call *Pretty Big*.

The easiest way to select the language is to use the Choose Language... menu item from the Language menu. The *Pretty Big* version can be found under the PLT submenu, which we expand by click on the triangle.

You may also be prompted to choose the language as soon as you start DrScheme for the first time. If that happens, you can find the *Pretty Big* version in the same place.

An Overview of The DrScheme User Interface

You will find that DrScheme has many similarities to most applications, including an Edit that supports operations like cut and paste. You may, however, notice that DrScheme has a primary window that looks somewhat different than other programs.



As you'll note, that window has two panes. The top pane is called the *Definitions Pane* and the bottom pane is called the *Interactions Pane*. As the names suggest, the top pane is used for writing definitions (more on those later), and the bottom pane is used for your primary interactions with DrScheme. We will start with the interactions pane.

For your initial use of DrScheme, you can treat the interactions pane as a fancy calculator. That is, you'll enter expressions and it will respond with the values of those expressions. For example, you can ask it to add 3 and 4, or to find the square root of 144.

Detour: A Bit of Scheme

The expressions you write in the interactions pane have to use the syntax of the Scheme programming language. Scheme uses a consistent syntax that makes a lot of sense to the computer, but that takes a bit for humans to master. (Fortunately, Scheme's syntax is much simpler than that of another language.)

Here are the two key points to remember when writing Scheme expressions:

- *Parenthesization*: Every expression and subexpression must be surrounded by parentheses. Hence, to ask for the square root of 144 we would write `(sqrt 144)`.
- *Prefix order*: Operations precede their operands. Hence, to add 3 and 4, we would write `(+ 3 4)` rather than the more traditional `3 + 4`.

Why does Scheme use these two key ideas? Parenthesization helps Scheme resolve potentially ambiguities. For example, you'll note that different calculators compute different results for $3+4*5$, depending on whether or not they are designed to accommodate precedence. In Scheme, since you must parenthesize every subexpression, you must write either `(+ 3 (* 4 5))` or `(* (+ 3 4) 5)`. Each expression indicates precisely what you want computed.

Prefix order, on the other hand, is intended to make life easier for the programmer. In particular, in languages in which some operations appear before their operands (such as `sqrt`) and others appear between their operands (such as `+` or `modulo`), programmers must reflect on which order to use for each operation. In Scheme, there's never a question, because you always put the operation first.

The Interactions Pane

When DrScheme is ready for you to type an expression in the pane, it prints a greater-than sign (also known as a right angle bracket). You type an expression you want evaluated and then hit the `<Return>` key. If you have not closed all the open parentheses, it will let you type more on the next line. Once you have completed an expression, it will evaluate it and present the result.

```
> (+ 3 4)
7
> (sqrt 144)
12
> (string-append "Hello" " " "Sam")
"Hello Sam"
```

While the first two expressions are numeric, the last shows that DrScheme can work with words, too.

Editing in the Interactions Pane

At times, you will find that you made a small typo in a long expression. For example, in computing the average of the grades 90, 67, 80, 85, 100, and 91, you might write

```
> (/ (+ 90 67 80 85 100 91) 5)
102 3/5
```

The result suggests, of course, that there was an error in the expression. (After all, the average of grades that are all 100 or less should not be more than 100.) A bit of analysis suggests that you really wanted to divide by 6, rather than 5. What can you do? You can retype the whole expression, but that takes a bit of time and gives you an opportunity to mistype something. You can cut and paste, and then edit the 5 (using the arrow keys and the backspace or delete keys).

Alternately, you can take advantage of DrScheme’s *Command History*. DrScheme lets you scan through the previous expressions entered. To back up, you enter <Esc> (which appears in the upper-left-hand corner of the keyboard) and then <P> for “Previous”. You can type <Esc> and <P> multiple times to back up more. You can also type <Esc> and <N> for “Next”.

The Definitions Pane

We’ve seen what the interactions pane is for, so what should we use definitions pane for? The simplest use is to assign names to values to make it easier to write our expressions. For example, if we wanted to work with grades on six homeworks, we might name the grade on the first homework `grade1`, the grade on the second homework `grade2`, and so on and so forth. That way, the expression to compute the average grade would be the clearer

```
> (/ (+ grade1 grade2 grade3 grade4 grade5 grade6) 6)
```

In Scheme, we name values using the `define` operation, which we put in the definitions pane.

```
(define grade1 90)
(define grade2 67)
(define grade3 80)
(define grade4 85)
(define grade5 100)
(define grade6 91)
```

The definitions in the definitions pane are evaluated when you click the Run button. Once you’ve done so, you can use the names in the interactions window.

Saving and Restoring Definitions

If you spend a bit of time naming values, you may find it useful to save those definitions into a file that you can then restore later. As you might guess, you save definitions using a menu item from the Save. Since there are two panes in the window, DrScheme provides both Save Definitions and Save Interactions (as well as a host of variations thereof). Since the definitions are more likely to be of permanent use than the interactions, the default is to save definitions.

The custom is to save Scheme files with a suffix of `.ss` or `.scm`.

Once you have saved definitions, you can quit DrScheme, go off and do other work, and then restart DrScheme and reload the definitions. As you might expect, you can load old definitions by using Open... or Open Recent

If you end up creating lots of definitions files and want to use them simultaneously (but don't need to edit them), there's a command you can type in the interactions window to get definitions from a file rather than from the definitions pane, `load`. You need to put the name of the file in quotation marks. For example,

```
> (load "mygrades.ss")
```

Concluding Thoughts

You have now learned enough to begin to interact with DrScheme. In a subsequent laboratory, we will ground these abstract instructions in concrete exercises.

Copyright © 2007 Samuel A. Rebelsky. 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.