Scheme Project
401-Programming Languages

1 Scheme Installation

The MIT/GNU Scheme development environment provides an interpreter, compiler, source-code debugger, integrated Emacs-like editor, and a large runtime library. MIT/GNU Scheme is available from http://www.gnu.org/software/mit-scheme/.

- Installation on OS X and Windows: Follow the instructions on the website.
- Installation on *nix: The MIT/GNU Scheme can be installed from the available source package using the GNU make utilities. If you do not have superuser privileges, you can specify an installation directory with the configure command (`configure --prefix=/INSTALLATIONDIR`)
- Installation on Debian based Linux distributions: MIT/GNU Scheme for x86 is also available in the software repository (package name `mit-scheme`).

2 Assignment

Compilers use various optimizations and optimization passes to improve the intermediate code representation. One such pass is algebraic simplification, which can be used on addresses or when the integer type is platform independent. The central idea of algebraic simplification is to generate sums of products. This plays well with many other optimization passes, such as constant propagation.

Algebraic simplification is a tree rewrite mechanism that traverses an AST in a bottom-up fashion and applies pattern matching to detect optimizable subtrees. If a pattern matches, then the sub-tree is rewritten according to the transformation specification.

In Scheme, compilers can represent arithmetic expressions in form of s-expressions. An s-expression is a recursive tree like data-structure comprised of lists. S-expressions have the form ($op\ exp_1\ exp_2$) where exp refers either to an atom (an integer constant or variable) or recursively to another s-expression. Implement algebraic simplification of s-expressions according to the rules outlined by Muchnick [1, 12.3.1]. Constants are denoted by c, other nodes (terms) t. A parenthesized expression, such as ($+\ c_1\ c_2$) represents an AST node (plus of constant 1 and constant 2). An unparenthesized expression, such as $c_1 + c_2$, means that the compiler can compute the result at compile time.

Note: Working through section 2.3 Symbolic Differentiation in `https://mitpress.mit.edu/sicp/full-text/book/book-Z-H-4.html` should be helpful.

Assignment report: Turn in the source code of your project together with an assignment report. The assignment report should contain: (1) names of two team members including a short statement of work, (2) a description of your design, (3) what was difficult, (4) what did you like about Scheme, (5) what did you dislike about Scheme.

$$
\begin{aligned}
(+\ c_1\ c_2) &\rightarrow c_1 + c_2 && (1)\\
(+\ t\ c) &\rightarrow (+\ c\ t) && (2)\\
(*\ c_1\ c_2) &\rightarrow c_1 * c_2 && (3)\\
(*\ t\ c) &\rightarrow (*\ c\ t) && (4)\\
(-\ c_1\ c_2) &\rightarrow c_1 - c_2 && (5)\\
(-\ t\ c) &\rightarrow (+\ (-c)\ t) && (6)\\
(+\ t_1\ (+\ t_2\ t_3)) &\rightarrow (+\ (+\ t_1\ t_2)\ t_3)) && (7)\\
(*\ t_1\ (*\ t_2\ t_3)) &\rightarrow (*\ (*\ t_1\ t_2)\ t_3)) && (8)\\
(+\ (+\ c_1\ t)\ c_2) &\rightarrow (+\ (+\ c_1\ c_2)\ t) && (9)\\
(*\ (*\ c_1\ t)\ c_2) &\rightarrow (*\ (*\ c_1\ c_2)\ t) && (10)\\
(*\ (+\ c_1\ t)\ c_2) &\rightarrow (+\ (*\ c_1\ c_2)\ (*\ c_2\ t)) && (11)\\
(*\ c_1\ (+\ c_2\ t)) &\rightarrow (+\ (*\ c_1\ c_2)\ (*\ c_1\ t)) && (12)\\
(*\ (+\ t_1\ t_2)\ c) &\rightarrow (+\ (*\ c\ t_1)\ (*\ c\ t_2)) && (13)\\
(*\ c\ (+\ t_1\ t_2)) &\rightarrow (+\ (*\ c\ t_1)\ (*\ c\ t_2)) && (14)\\
(*\ (-\ t_1\ t_2)\ c) &\rightarrow (-\ (*\ c\ t_1)\ (*\ c\ t_2)) && (15)\\
(*\ c\ (-\ t_1\ t_2)) &\rightarrow (-\ (*\ c\ t_1)\ (*\ c\ t_2)) && (16)\\
(*\ (+\ t_1\ t_2)\ t_3) &\rightarrow (+\ (*\ t_1\ t_3)\ (*\ t_2\ t_3)) && (17)\\
(*\ t_1\ (+\ t_2\ t_3)) &\rightarrow (+\ (*\ t_1\ t_2)\ (*\ t_1\ t_3)) && (18)\\
(*\ (-\ t_1\ t_2)\ t_3) &\rightarrow (-\ (*\ t_1\ t_3)\ (*\ t_2\ t_3)) && (19)\\
(*\ t_1\ (-\ t_2\ t_3)) &\rightarrow (-\ (*\ t_1\ t_2)\ (*\ t_1\ t_3)) && (20)
\end{aligned}
$$

References

[1] Steven S. Muchnick. *Advanced compiler design and implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.