TETE: Unit Testing for Source Transformation

Derek M. Shimozawa, James R. Cordy, and Adrian Thurston
Software Technology Laboratory, School of Computing, Queen's University
Kingston, Canada

I. Motivation

Debugging Transformation Rules

Source transformation processes are implemented using sets of individual subtransformations that map syntactic substructures in the problem domain to those of the solution domain. Typically, rewrites cannot be executed independently from the overall transformation. As a result, localizing and analyzing rewrite errors within a large group of rules can be difficult, especially for inexperienced users [4].

Debugging Application Strategies

Rewrite systems also include traversal control facilities and application conditions to define the order and scope of application of intermediate tree rewrite rules. Determining an appropriate application strategy can be a challenging task, in part because rewrites can be composed in various arrangements to return syntactically valid, but semantically incorrect, results.

Debugging Transformation Grammars

Transformation systems require the definition of a context-free grammar that describes the form of the source input. Transformation grammars often allow for unrestricted grammatical forms, without analyzing or checking for errors. Even though a grammar may be flawed, an erroneous definition often will not surface until the user generates a parse that descends into the flawed structure.

Pedagogy

Source transformation systems usually work well to help experienced developers quickly produce source transformations, at the cost that they presume a strong grasp of the underlying paradigm fundamentals and place their target audience well above the introductory level [1].

II. TETE

A Unit Testing Framework

To address the unique learning and development challenges posed by the source transformation paradigm, we present the TXL Engineering Toolkit for Eclipse (TETE). TETE is a set of **Eclipse** plug-ins that provides a simple, consistent interface for automatically and non-invasively unit testing pattern replacements (subtransformations), rewrite strategies (which are implicitly defined by the composition of subtransformations), and grammar types written in the TXL source transformation language [2, 6].

III. Rule Testing

Pattern-Replacement Isolation

Structurally, TXL rules are organized into a rooted pure functional program in which lower level rules are applied as functions on scopes captured by higher level patterns. By synthesizing a transformation entry point that directly invokes the rule of interest (*target rule*), a user can bypass the rules that reside outside of the target rule's scope, thereby isolating the pattern-replacement effects of the target rule.

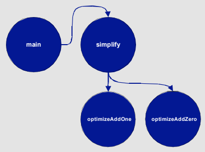

Fig. 1. *TXL function call graph*

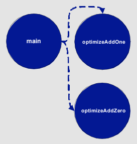

Fig. 2. *Call graph for modified TXL program*

Application Strategy Decomposition

Rule application strategies are implicitly programmed as part of the functional decomposition of the transformation rule set, which controls how and in which order subrules are applied. Transformation rule isolation breaks down a strategy into isolated regions that can be evaluated separately. The result is a new application strategy with the target rule acting as the entry point.

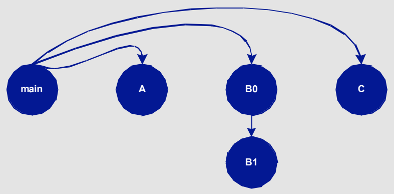

Fig. 3. *Unmodified TXL rule application strategy*

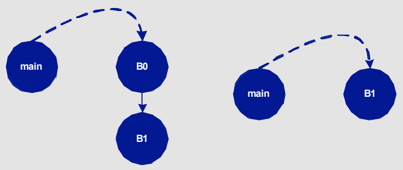

Fig. 4. *Application strategy fragments*

IV. Grammar Testing

Context-free Grammar Type Isolation

TXL Grammar redefinitions (*overrides*) can be used for the generation of language dialects and variants without modifying the base grammar. The *effective grammar* is the one that is formed by substituting each redefinition into the base grammar in the order that it appears in the TXL program. TETE automatically isolates a non-terminal of interest (*target type*) by redefining the root non-terminal so that it directly references the target type.

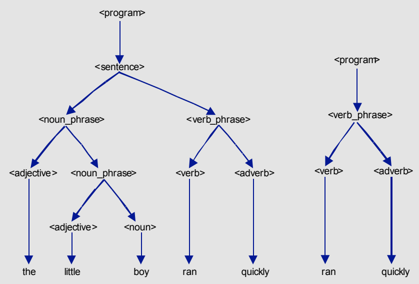

Fig. 5. *TXL grammar type isolation*

Grammar Scope Refinement

When used together with rule isolation, type isolation greatly simplifies the evaluation process by refining a program's grammar according to the specific scoping requirements of the target rule. Programmers can therefore test their rules with refined instances of grammatical input that directly and concretely demonstrates pattern-replacement behavior.

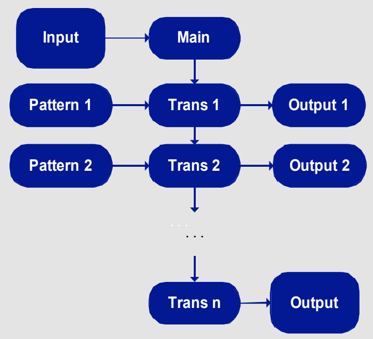

Fig. 6. *Refinement of scope to rule patterns*

V. Unit Test Automation

Thus far, our methodology has been focused on using functional decomposition and grammar overriding to isolate individual TXL rules and non-terminals [3]. Without an automation process, a program must be manually mutated to denote new rule and grammar entry points. To address this issue, the TXL Test Generator provides non-invasive test stub generation to automate the program mutation process for TXL programmers [5].

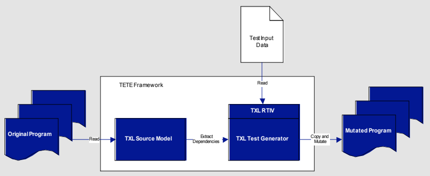

Fig. 8. *TXL Test Generator*

VI. Conclusion

TETE provides a set of Eclipse plug-ins designed to make editing, testing and debugging TXL source transformations simpler and more accessible. As the range of transformation applications increases, the role of TETE as a general solution environment will become a compelling possibility for the future.

References

[1] E. Allen, R. Cartwright, and B. Stoler. DrJava: A Lightweight Pedagogic Environment for Java. *SIGCSE Bull.*, pp. 137-141, 2002.

[2] J. R. Cordy. TXL – a language for programming language tools and applications. *Proc. LDTA 2004,* pp. 1-27, 2004.

[3] S. H. Edwards. Rethinking Computer Science Education from a Test-First Perspective. In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications*, pp. 148-155, 2003.

[4] R. B. Findler, J. Clements, C. Flanagan, M. Flatt, S. Krishnamurthi, P. Steckler, and M. Felleisen. DrScheme: A Programming Environment for Scheme. *Journal of Functional Programming*, pp. 159-182, 2002.

[5] OTI Technologies. Eclipse Platform Technical Overview. Technical report, Object Technology International, Inc., February 2003. http://www.eclipse.org/whitepapers/eclipse-overview.pdf; accessed 2004

[6] C. Reis and R. Cartwright. A Friendly Face for Eclipse. In *Eclipse '03: Proceedings of the 2003 OOPSLA workshop on Eclipse technology eXchange*, pp. 25-29, 2003.

Presented at CASCON 2005 , Richmond Hill, Ontario, Canada (October 2005)

This work was supported by an IBM Eclipse Innovation Grant and by the Natural Sciences and Engineering Research Council of Canada