Prokopis Georgiou

COMP2209 - Coursework Challenges

Report on Development and Testing Strategy

The solutions for Challenges 1 to 6 were developed incrementally, emphasizing modularity, recursion, and strong type safety. Challenge 1 focused on calculating ray interactions in a grid using recursive functions and custom data types like EdgePoint and Face. The movement and reflection logics were modularized, ensuring clarity and reducing redundancy. Testing involved grids of various sizes, validating edge cases like empty grids or rays reflecting on entry, with debug tools such as trace aiding in troubleshooting.

Challenge 2 builds upon Challenge 1 by solving for atom placements based on ray interactions. It used combination generation and backtracking to identify valid configurations. Testing verified functionality across different scenarios, including single and multiple atoms, empty grids, and cases with multiple valid solutions. Sorted outputs helped compare results for non-deterministic cases.

Challenge 3 tackled parsing and unparsing of lambda expressions, introducing custom types like LamExpr and LamMacroExpr. The unparsing function transformed internal representations into human-readable forms while maintaining syntactic correctness. Testing covered cases from simple expressions to nested applications and macro usage, ensuring that the parser could handle lambda-calculus intricacies.

Challenge 4 extended parsing to include macros, implementing validation for well-formed macro definitions and closed expressions. The tokenizer handled various input formats, while recursive parsing logic supported nested expressions and chains of applications. Testing ensured correct handling of valid macros and rejection of invalid constructs, such as mismatched parentheses or nested macro definitions.

Challenge 5 involved transforming lambda expressions into Continuation Passing Style (CPS). This required recursive transformations while maintaining semantic equivalence. The testing compared transformed expressions with manually derived CPS forms, ensuring correctness for simple and nested abstractions, macros, and applications.

Challenge 6 focused on beta reduction, comparing inner and outer reduction strategies. Recursive functions for substitution and reduction supported both innermost and outermost evaluation. Testing measured reduction steps for various expressions, including terminating and non-terminating cases, verifying results against theoretical expectations.

The primary tools used included GHCi for interactive testing, property-based testing for generalized validation, and Debug.Trace for debugging recursive logic. Modular development improved maintainability, while systematic testing ensured robustness. This approach was effective in providing reliable and efficient solutions aligned with the assessment's requirements.