1. **Як переконатися в наявності чи відсутності конфліктів у парсері?**

Bison автоматично генерує звіт про конфлікти у файлі *src/parser/bison-report.txt*. Звіт містить детальну інформацію про конфлікти shift/reduce та reduce/reduce. У звіті вказується стан автомату, де виник конфлікт, та правила, що конфліктують.

Опція *--trace-parser* активує трасування парсера. Коли користувач запускає команду *src/driver/dtiger --trace-parser test.tig*, він отримує детальну інформацію про дії парсера.

Звіт Bison допомагає виявити потенційні проблеми в граматиці, а трасування допомагає побачити, як ці проблеми проявляються при аналізі конкретних виразів.

2. **Зміст %prec UMINUS**

Директива *%prec* використовується, коли нам потрібно вказати пріоритет для правила, який відрізняється від стандартного пріоритету символів у цьому правилі. Особливо корисно для унарних операторів, які використовують такий же символ, що й бінарні.

Це означає, що правило «унарний мінус» матиме пріоритет, заданий для *UMINUS*, а не стандартний пріоритет знака мінус «-», який зазвичай використовується для віднімання.

Символ «-» використовується як для унарного мінуса (-4), так і для віднімання (4-5). Ці дві операції мають різні пріоритети в математиці. Без *%prec UMINUS* парсер не зможе правильно інтерпретувати вирази типу -4+5.

3. **Відмінність else**

else return yy::tiger_parser::make_ELSE(loc);

Це частина **лексера**, де визначаються дії для розпізнаних лексем. Коли лексер знаходить у вхідному коді слово *«else»*, він викликає функцію *make_ELSE(loc)*, яка створює токен *ELSE* з інформацією про його розташування в коді *(loc)*, і повертає цей токен парсеру.

Ця частина відповідає за фактичне розпізнавання ключового слова *«else»* у вхідному тексті та перетворення його в токен для подальшої обробки парсером.

%token
...
ELSE "else"

Це частина **парсера**, де оголошуються токени, які парсер буде отримувати від лексера. Тут *ELSE* визначається як токен, а *«else»* - це

літеральне представлення цього токена, яке використовуватиметься в повідомленнях про помилки та документації.

Цей фрагмент не містить логіки для розпізнавання *«else»* - він лише декларує, що в граматиці мови Tiger існує такий токен, і парсер повинен знати, як його обробляти відповідно до правил граматики.

4. Evaluator принцип роботи

ASTEvaluator реалізує принцип роботи через патерн відвідувача (*Visitor Pattern*) для обчислення значень виразів у абстрактному синтаксичному дереві (*AST*). *ASTEvaluator* є нащадком класу *ConstASTIntVisitor*, що дозволяє йому «відвідувати» різні типи вузлів *AST* і обчислювати їх числові значення.

Кожен метод *visit* приймає конкретний тип вузла *AST*. Для складних виразів (наприклад, бінарних операторів) рекурсивно обчислюються значення підвиразів через виклик *accept(*this)*.

- **IntegerLiteral:** просто повертає числове значення.
- **BinaryOperator:** обчислює вирази зліва та справа і застосовує відповідну операцію (+, -, *, /, ==, !=, тощо).
- **Sequence:** обчислює всі вирази послідовно і повертає результат останнього.
- **IfThenElse:** обчислює умову і, залежно від результату, виконує then або else частину.

Для невпроваджених типів вузлів (*String*, *Let*, *Identifier*, тощо) генерується помилка *«Not implemented»*. Також обробляються особливі випадки, як-от порожня послідовність.