20. Использование неоднозначных грамматик в LR-анализе. Примеры неоднозначностей, разрешимых при LR-анализе. Обработка синтаксических ошибок.

Использование **неоднозначных грамматик** в LR-анализе является распространенной практикой, так как это позволяет создавать более компактные и быстрые синтаксические анализаторы. Несмотря на то, что для работы компилятора необходимо однозначное описание языка, на практике часто удобнее использовать неоднозначную грамматику, дополненную специальными «внеграмматическими» правилами для разрешения возникающих конфликтов.

Использование неоднозначных грамматик и их преимущества

Применение неоднозначных грамматик дает несколько преимуществ:

- **Уменьшение размера:** количество состояний в LR-автомате сокращается.
- **Повышение скорости:** анализатор не тратит время на свертки по «цепным правилам» (например, $E \to T$ и $T \to F$), которые обычно вводятся в однозначных грамматиках только для отражения приоритетов.

Примеры разрешимых неоднозначностей

В LR-анализе неоднозначность проявляется в виде **конфликтов перенос-свертка** или **свертка-свертка**, которые разрешаются вручную на стадии построения таблиц.

1. **Арифметические выражения:**
   Для грамматики $E \to E + E \mid E * E \mid (E) \mid x$ конфликты разрешаются на основе **приоритета и ассоциативности** операторов.
   - Если приоритет входного символа выше (например, в стеке «+», а на входе «*»), выбирается **перенос**.
   - Если приоритеты равны, выбор зависит от ассоциативности: для левоассоциативных операций (как «+») выполняется **свертка**, чтобы выполнить левую операцию первой.
2. **Проблема «висячего else»:**
   В конструкциях `if...then...else` возникает конфликт: к какому `if` отнести текущий `else`. Общепринятый принцип — **else всегда относится к ближайшему if**. В LR-автомате этот конфликт перенос-свертка разрешается в пользу **переноса** символа `else` в стек.

Обработка синтаксических ошибок

LR-анализатор обнаруживает ошибку, как только текущий входной символ не может корректно продолжить активный префикс в стеке. Технически это происходит при обращении к **пустой клетке таблицы action**.

Основные стратегии обработки:

- **Процедуры исправления:** В пустые клетки помещаются ссылки на подпрограммы, которые могут «исправить» ситуацию: например, вставить пропущенный операнд или оператор, либо удалить лишнюю скобку.
- **Предварительные свертки:** Чтобы добавить исправляющий символ в стек (например, операнд `x`), анализатор иногда должен сначала выполнить все возможные свертки, чтобы стек снова содержал корректный активный префикс.
- **Режим паники (базовая стратегия):** Пропуск символов во входном потоке до тех пор, пока не встретится терминал, позволяющий продолжить анализ (обычно это символы из множества $follow$ для текущего нетерминала).

Главным достоинством LR-анализа в этой области является **быстрота реагирования**: ошибка фиксируется немедленно, и ошибочный символ даже не переносится в стек.