В процессе обхода синтаксического дерева, может быть 2 ситуации:

1. L:

GOTO L;

1. GOTO L;

L;

Ситуация №1 банальна, при встрече метки она добавляется в стек меток. Затем при встрече GOTO происходит поиск встреченной метки в стеки.

Ситуация №2 Намного сложнее, тем что может быть такой вариант использование:

1. {

GOTO L;

}

L;

То есть здесь происходит выход из внутреннего блока в родительский.

Но для начала необходимо продолжить рассмотрение ситуации №2, затем реализация ситуации №3 станет очевиднее.

В ситуациях (№2, №3), когда используемая метка внутри оператора GOTO не найдена, то она добавляется в таблицу ожидаемых меток. Только лишь после полного обхода метода, можно судить “Были ли найдены все заявленные метки”. При входе в очередной блок происходит запоминание текущей таблицы и её сброс затем после обхода блока происходит восстановления прежней таблицы и добавления новой таблицы в родительскую. Это необходимо для реализации ситуации №3. (Так как оператор перехода, может ссылаться на метки верхнего уровня). При детальном рассмотрении, таблица сохраняет не непосредственно метки а операторы “GOTO”, которые не были удовлетворены на данный момент. Как уже было сказано, при входе в новый блок происходит сброс таблицы, так как:

1. GOTO L;

{

L:

}

Такой переход не возможен и значит в новом блоке не стоит искать уже заявленные метки.