**5. Процесс верификации класса Java**

Правило Гослинга — размер стека и тип переменных не зависит от пути в графе управления.

Граф управления — все возможные варианты реализаци ветвлений в программе.

Граф потока данных — для каждой операции есть набор входных данных и набор выходных данных.

Верификация

* Проверка формата класса
* Проверка данных класса
  + final классы и методы не перекрыты
  + Определен суперкласс
  + Формат пула констант
* Проверка кода класса
  + Размер стека, тип данных на стеке
  + Правильный тип локальной переменной при доступе
  + Число аргументов при вызове метода
  + Необходимое количество и тип аргументов инструкций
* Проверка ссылок
  + Наличие классов, методов и полей
  + Права доступа

По стандарту проверка ссылок осуществляется лениво.

Алгоритмически задача верификации неразрешима за конечное время.

[https://blogs.oracle.com/vmrobot/entry/%D0%B2%D0%B5%D1%80%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F\_%D0%B1%D0%B0%D0%B9%D1%82\_%D0%BA%D0%BE%D0%B4%D0%B0\_%D0%B2\_jvm](https://blogs.oracle.com/vmrobot/entry/верификация_байт_кода_в_jvm)

Верификатор хранит состояние локальных переменных и стека в каждой точке программы. Когда некоторая инструкция программы выполняется еще раз, сохраненное состояние объединяется с текущим, а разные типы заменяются общим супертипом.

Затем, если инструкция загружает переменную или значение из стека, ожидаемый тип сравнивается с действительным. Если действительный тип не может быть присвоен ожидаемому типу (т.е., не является тем же типом или потомком ожидаемого типа в дереве), класс считается невалидным.

Таким образом, верификатор пытался определить типы, имитируя выполнение программы. В худшем случае, сложность верификации была O(n \\* log2 t), где t — количество типов в дереве типов (включая классы Java), а n — количество инструкций.