Введем некоторые определения:

- $at(\phi)$ - множество атомов в формуле ϕ
- $at_i(\phi)$ - некоторый заданный порядок этих атомов
- Атом a сопоставим с булевой переменной $e(a)$ (такую процедуру будем называть булевское кодирование)
- $e(t)$ - булева формула, полученную булевым кодированием каждого атома формулы t (такую формулу будем называть пропозиционным скилетом формулы t)

Рассмотрим следующий пример: $\phi := x = y \vee x = z$. Тогда $at(\phi) = \{at_1, at_2\}$, $at_1 := x = y$, $at_2 := x = z$. Сопоставим $x = y$ булеву переменную $e(x = y) = b_1$ и $x = z$ булеву переменную $e(x = z) = b_2$. Получаем $e(\phi) = b_1 \vee b_2$.
Пусть α - некоторая (возможно, частичная) оценка формулы $e(\phi)$, например для формулы выше пусть $\alpha = \{e(at_1) \to FALSE, e(at_2) \to TRUE\}$.

$$Th(at_i, \alpha) = \begin{cases} at_i & \text{если } \alpha(at_i) = TRUE \\ \neg at_i & \text{иначе} \end{cases}$$

$Th(\alpha) = \{Th(at_i, \alpha) | e(at_i), \alpha\}$, $\overline{Th(\alpha)}$ - конъюнкция всех элементов их $Th(\alpha)$
Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$. Опишем алгоритм решение SMT-формул, имея *Deduction*:

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили некоторую α, то вычисляем *Deduction* от $\overline{Th(\alpha)}$
- Если получили "SAT то возвращаем "SAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили "UNSAT то $B = B \wedge$ "блокирующий дизюнкт"t и начинаем со второго пункта

Опишем подробнее свойства "блокирующего дизюнкта"(так же называют леммой):

- t - тавтология в T
- t - состоит только из атомов из ϕ
- t - "блокирует"α, т.е. при отправлении B SAT-решателю мы не сможем снова получить α

Предположим, что в предыдущем примере оценка оказалась "UNSAT тогда можно привести такой "блокирующий дизюнкт": $t := e(at_1) \vee \neg e(at_2)$ (т.е. мы просто взяли дизъюнкцию отрицаний атомов). Есть и другие способы найти блокирующий дизюнкт, например, часто можно взять дизъюнкцию части отрицаний атомов, если они "UNSAT"в данной теории.

Есть более эффективные алгоритмы решения SMT задач, но для этого нам нужно изучить, как работает SAT решатель. В процессе работы ленивого алгоритма может возникнуть много блокирующих дизюнктов. Многие теории можно свести к SAT задаче, но обычно существуют более эффективные алгоритмы решения теорий.

Решим с помощью SMT-решателя задачу с прошлой лекции: задача о расскарске графа. Имеется граф $G = (V, E)$. Можно ли его вершини расскрасить в k цветов так, чтобы никакие соседние вершины не были раскрашены в один и тот же цвет? Постороим следующую систему уравнений:

- x_i - целые
- $1 \leq x_i \leq c$
- $x_i \neq x_j$ для $(x_i, x_j) \in E$

Для решения такой системы нам понадобится SMT-решатель, умеющий разрешать теорию равенства.
В качестве упражнения предоставляются следующие задачи:

- Судоку (заполнить поле числами)
- Сапер (указать, на какую клетку можно "кликнуть")

В конце лекции разбрем следующую задачу - даны две программы, a и b:
Программа a:
int i, out_a = in;
for (int i = 0; i < 2; ++i) out_a = out_a * i;
return out_a;
Программа b:
int out_b = (in * in) * in;
return out_b;
Нужно проверить, являются ли они эквивалентными друг другу. Для этого построим формулы, описывающие связь между входом и выходом программ:
$\phi_a := (out\_a_0 = in\_a_0) \wedge (out\_a_1 = out\_a_0 * in\_a_0) \wedge (out\_a_2 = out\_a_1 * in\_a_0)$
$\phi_b := out\_b_0 = (in\_b_0 * in\_b_0) * in\_b_0$
Чтобы программы были эквивалентны, должно выполняться:
$(in\_a_0 = in\_b_0) \wedge \phi_a \wedge \phi_b \rightarrow out\_a_2 = out\_b_0$
Проверку такой формулы на выполнимость может осуществить SMT-решатель, умеющий разрешать теорию равенства.