Проверка выполнимости: SAT-решатели

Лекция №5

Со времен греков говорить «математика» – значит говорить «доказательство».

Н. Бурбаки. Элементы математики

Александр Сергеевич Камкин

kamkin@ispras.ru

Инструменты: пруверы, солверы и т.п.

- Системы доказательства теорем (provers)
- Системы проверки выполнимости (solvers)
- Системы помощи в доказательстве (proof assistants)
- Системы проверки доказательств (proof checkers)

Общезначимость и выполнимость

- Выполнимость – существование модели
- Общезначимость – истинность во всех интерпретациях
- Формула φ общезначима $\Leftrightarrow$ $\neg\varphi$ невыполнима (UNSAT)
- Формула φ выполнима (SAT) $\Leftrightarrow$ $\neg\varphi$ необщезначима

Логика высказываний: КНФ-выполнимость

- **Вход: КНФ (клаузальная форма)**
  - Формула – множество клауз
  - Клауза (дизъюнкт) – множество литералов
  - Литерал (буква) – атом или его отрицание
  - Атом – элементарное высказывание
- **Примеры клаузальных форм**
  - $\emptyset \equiv true$ — пустая формула
  - $\{\square\} \equiv false$ — формула, состоящая из пустой клаузы
  - $\{pr, \bar{q}\bar{p}q, p\bar{p}q\} \equiv \big\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\big\}$

Кодировка Цейтина: формула $\rightarrow$ КНФ

- Эквивалентные преобразования к КНФ – тупик!
  - Размер КНФ экспоненциально зависит от размера ДНФ
- Кодировка Цейтина (1968)
  - Получаемая КНФ <u>равносильна</u> исходной формуле
  - Могут потребоваться дополнительные переменные

Г.С. Цейтин
(1936-2022)

$$encode(\varphi) \equiv p_\varphi \wedge link(\varphi, p_\varphi)$$
$$link(\psi \circ \chi, p_{\psi\circ\chi}) \equiv cnf\left(p_{\psi\circ\chi} \leftrightarrow (p_\psi \circ p_\chi)\right) \wedge link(\psi, p_\psi) \wedge link(\chi, p_\chi)$$
$$link(\neg\psi, p_{\neg\psi}) \equiv link(\psi, \bar{p}_{\neg\psi})$$
$link(\gamma, p_\gamma) \equiv true$, если γ — литерал (в этом случае $p_\gamma \equiv \gamma$)

Кодировка Цейтина: базовые преобразования

Type	Operation	CNF Sub-expression
 AND	$C = A \cdot B$	$(\overline{A} \vee \overline{B} \vee C) \wedge (A \vee \overline{C}) \wedge (B \vee \overline{C})$
 NAND	$C = \overline{A \cdot B}$	$(\overline{A} \vee \overline{B} \vee \overline{C}) \wedge (A \vee C) \wedge (B \vee C)$
 OR	$C = A + B$	$(A \vee B \vee \overline{C}) \wedge (\overline{A} \vee C) \wedge (\overline{B} \vee C)$
 NOR	$C = \overline{A + B}$	$(A \vee B \vee C) \wedge (\overline{A} \vee \overline{C}) \wedge (\overline{B} \vee \overline{C})$
 NOT	$C = \overline{A}$	$(\overline{A} \vee \overline{C}) \wedge (A \vee C)$
 XOR	$C = A \oplus B$	$(\overline{A} \vee \overline{B} \vee \overline{C}) \wedge (A \vee B \vee \overline{C}) \wedge (A \vee \overline{B} \vee C) \wedge (\overline{A} \vee B \vee C)$
 XNOR	$C = \overline{A \oplus B}$	$(\overline{A} \vee \overline{B} \vee C) \wedge (A \vee B \vee C) \wedge (A \vee \overline{B} \vee \overline{C}) \wedge (\overline{A} \vee B \vee \overline{C})$

Кодировка Цейтина: пример $(p \oplus q) \to \bar{r}$

- Новая переменная $f \leftrightarrow \big((p \oplus q) \to \bar{r}\big)$
- Новая переменная $h \leftrightarrow (p \oplus q)$
- $h \leftrightarrow (p \oplus q)$
  - $(\bar{p} \vee \bar{q} \vee \bar{h}) \wedge (\bar{p} \vee q \vee h) \wedge (p \vee \bar{q} \vee h) \wedge (p \vee q \vee \bar{h}) \equiv \{\bar{p}\bar{q}\bar{h}, \bar{p}qh, p\bar{q}h, pq\bar{h}\}$
- $f \leftrightarrow (h \to \bar{r}) \Leftrightarrow f \leftrightarrow (\bar{h} \vee \bar{r})$
  - $(\bar{h} \vee \bar{r} \vee \bar{f}) \wedge (h \vee f) \wedge (r \vee f) \equiv \{\bar{h}\bar{r}\bar{f}, hf, rf\}$
- $\{f, \bar{h}\bar{r}\bar{f}, hf, rf, \bar{p}\bar{q}\bar{h}, \bar{p}qh, p\bar{q}h, pq\bar{h}\}$

Метод резолюций для логики высказываний

Вход: клаузальная форма F, не содержащая клаузы □

Выход: *true* (F выполнима) или *false* (F невыполнима)

while F содержит «неспаренные» сталкивающиеся клаузы **do**

  выбрать из F новую пару сталкивающихся клауз C_1 и C_2;

  $C := Res(C_1, C_2)$;

  if $C = □$ **then**

    return *false*

  end;

  $F := F \cup \{C\}$;

end;

$$Res(C_1, C_2) = (C_1 \setminus \{l\}) \cup (C_2 \setminus \{l^C\})$$

return *true*

Метод резолюций: пример

$$p \wedge (\neg p \vee q) \wedge \neg r \wedge (\neg p \vee \neg q \vee r) \equiv \{p, \bar{p}q, \bar{r}, \bar{p}\bar{q}r\}$$

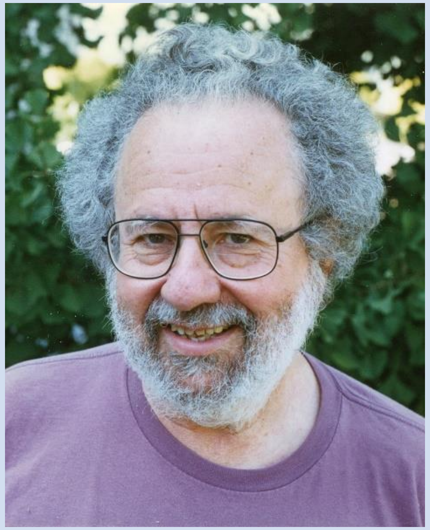
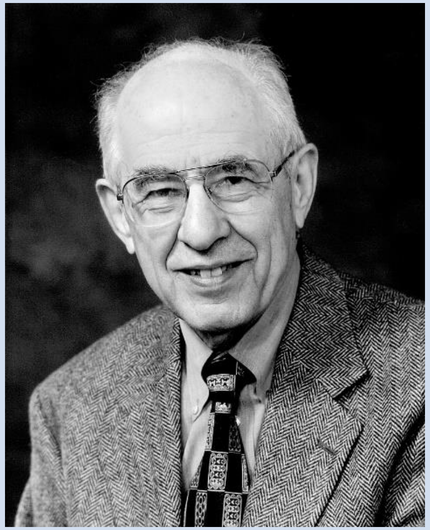
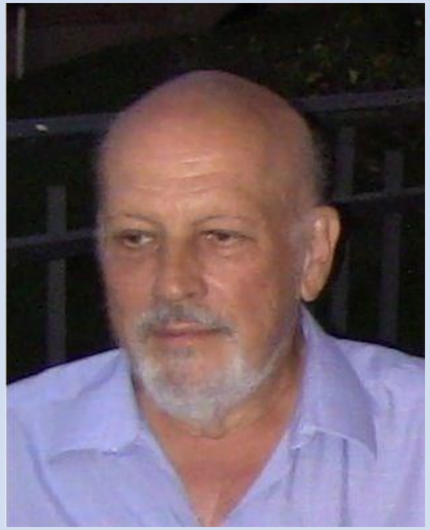
- $C_1 = p;$
- $C_2 = \bar{p}q;$
- $C_3 = \bar{r};$
- $C_4 = \bar{p}\bar{q}r;$
- $C_5 = Res(C_3, C_4) = Res(\bar{r}, \bar{p}\bar{q}r) = \bar{p}\bar{q};$
- $C_6 = Res(C_2, C_5) = Res(\bar{p}q, \bar{p}\bar{q}) = \bar{p};$
- $C_7 = Res(C_1, C_6) = Res(p, \bar{p}) = \square.$

UNSAT

Алгоритм DPLL (1962)

- Удаление тавтологий
  - Удаляются все клаузы, содержащие контрарные пары
- Распространение единицы
  - Если есть единичная клауза $\{l\}$
    - Удаляются все клаузы, содержащие литерал l
    - Из оставшихся удаляются вхождения контрарных литералов l^C
- Исключение чистых литералов
  - Если l – чистый литерал, т.е. литерал, входящий с одним «знаком»
    - Удаляются все клаузы, содержащие литерал l

DPLL = Davis + Putnam + Logemann + Loveland

D	P	L	L
Мартин Дэвис (1928-2023)	**Хилари Патнэм** (1926-2016)	**Джордж Логеман** (1938-2012)	**Дональд Лавленд** (род. 1934)
			

Алгоритм DPLL: псевдокод

```
Функция DPLL
Вход:  КНФ F
Выход: true ⇔ F выполнима
  F′ := удалить_тавтологии(F);
  DPLL′(F′, ∅)

Функция DPLL′
Вход:  КНФ F, частичная интерпретация I
Выход: true ⇔ F выполнима

F′ := F;
I′ := I;

while F′ содержит единичные клаузы do
  выбрать единичную клаузу {p^x};
  F′ := распространить_единицу(F′, p^x);
  I′ := I′[p := x]
end;

while F′ содержит чистые литералы do
  выбрать чистый литерал p^x;
  F′ := исключить_чистый_литерал(F′, p^x);
  I′ := I′[p := x]
end;

// конфликт
if □ ∈ F′ (F′ содержит клаузу, ложную в I′) then
  return false;
end;

// решение
if F′ = ∅ (все клаузы F′ истинны в I′) then
  return true;
end;

выбрать элементарное высказывание p, входящее в F′;
выбрать значение истинности x ∈ {true, false};

// вычисления выполняются по правилам короткой логики
return DPLL′(F′[p := x],  I′[p := x]) ∨
       DPLL′(F′[p := ¬x], I′[p := ¬x]);
```

Обучение на основе конфликтов (CDCL, 1996)

- Если формула φ ложна, находится причина конфликта
  - Множество присваиваний $x_{i_1} \leftarrow \sigma_{i_1}, \dots, x_{i_k} \leftarrow \sigma_{i_k}$
  - Соответствует конъюнкции $K = x_{i_1}^{\sigma_{i_1}} \wedge \cdots \wedge x_{i_k}^{\sigma_{i_k}}$
- В будущем нужно избегать таких присваиваний
  - Создаем клаузу $\overline{K} = x_{i_1}^{\overline{\sigma_{i_1}}} \vee \cdots \vee x_{i_k}^{\overline{\sigma_{i_k}}}$
- Добавляем клаузу $\overline{K}$ в формулу φ

Эвристика VSIDS (Variable State Independent Decaying Sum)

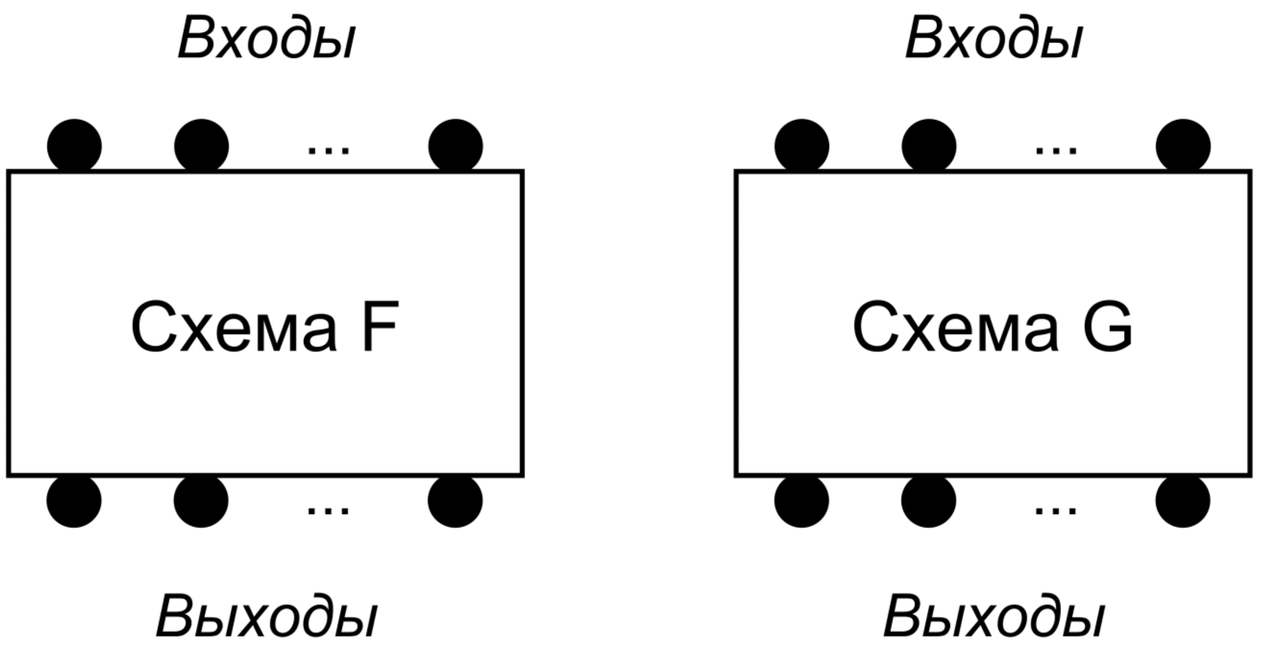
- В начале работы решателя *активность* каждого литерала устанавливается равной 0
- При выводе клаузы (и при загрузке) активность входящих в клаузу литералов увеличивается на 1 (*additive bump*)
- Через регулярные интервалы времени активность всех литералов умножается на $\alpha \in (0,1)$ (*multiplicative decay*)
- Для ветвления выбирается литерал с максимальной активностью

Стохастические решатели GSAT и WalkSAT

- Присвоить переменным случайные значения
- Пока φ содержит ложные клаузы
  - **GSAT**: выбрать переменную x
    - изменение x минимизирует <u>число ложных клауз</u>
    - иногда (с небольшой вероятностью) – случайная переменная
  - **WalkSAT**: выбрать переменную x
    - случайно выбрать ложную клаузу, потом – переменную в ней
    - изменение x минимизирует <u>число истинных клауз, которые станут ложными</u>
    - если таких переменных несколько, выбрать случайно
  - $x \leftarrow \overline{x}$
  - Если решение не находится долго
    - Перезапуск с новыми случайными значениями

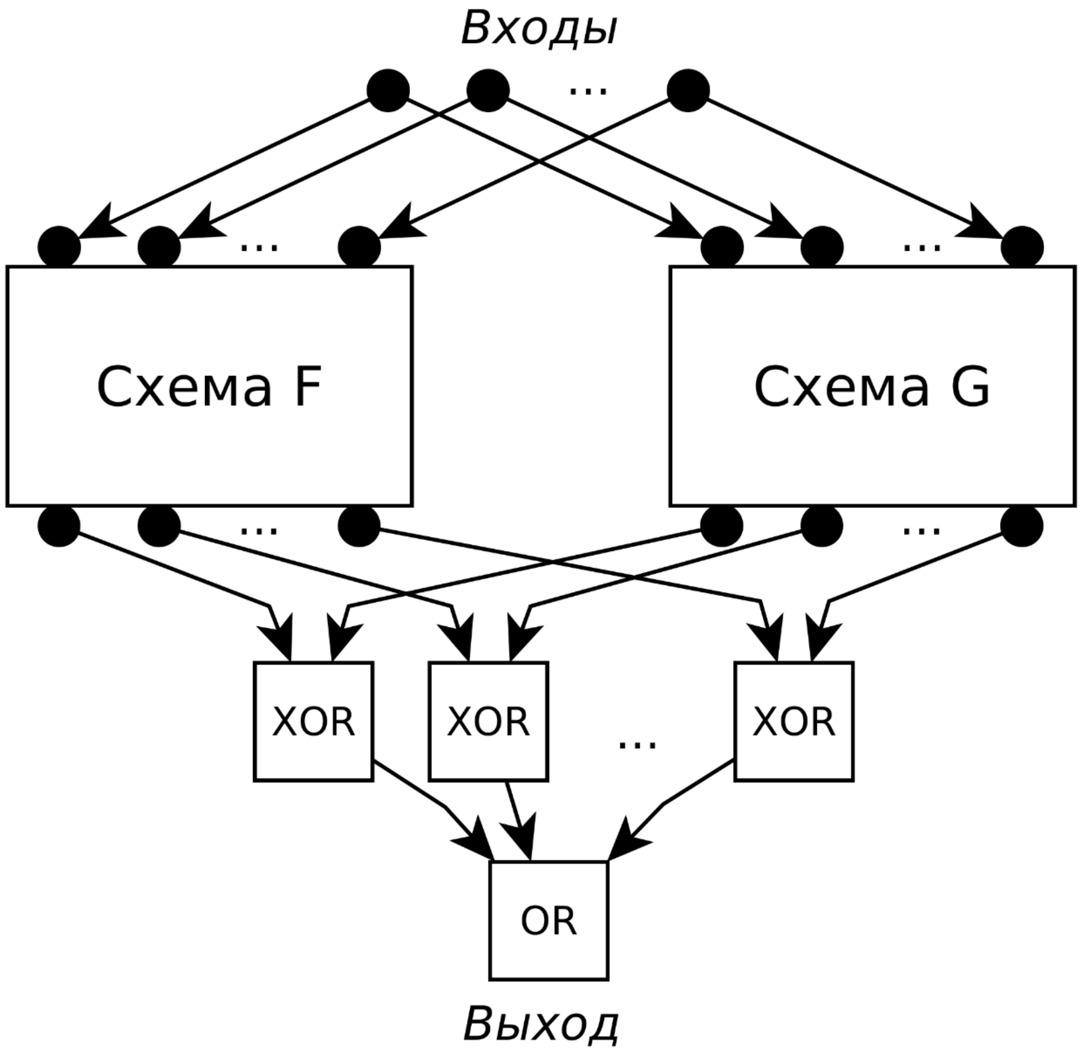
Применение SAT-решателей к LEC

- **Даны две схемы**
  - Соответствие входов
  - Соответствие выходов
- **Выяснить**
  - Эквивалентны ли схемы
    - Если нет
      - *Контрпример*
      - *Диагностика*



$$\forall \alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{B}^n\colon\ F(\alpha) = G(\alpha)$$

Схема-митра (miter) $M[F, G]$



- **Эквивалентность $\boldsymbol{F}$ и $\boldsymbol{G}$:**

  $\forall \alpha \in \mathbb{B}^n\colon M[F, G](\alpha) = 0$

  Невыполнимость (UNSAT)

- **Неэквивалентность $\boldsymbol{F}$ и $\boldsymbol{G}$:**

  $\exists \alpha \in \mathbb{B}^n\colon\ M[F, G](\alpha) = 1$

  Выполнимость (SAT)

Замечание: выбор названия становится понятным, если схему перевернуть: сверху изобразить выход, снизу – входы

Базовый алгоритм проверки эквивалентности

- Построить схему-митру $M[F, G]$
- Построить кодировку Цейтина $\varphi_{F,G} \equiv encode(M[F, G])$
- Проверить выполнимость $\varphi_{F,G}$ с помощью SAT-решателя
  - Если $\varphi_{F,G}$ выполнима (SAT), схемы F и G неэквивалентны
    - Выдать контрпример
  - Иначе схемы F и G эквивалентны

Итеративная проверка эквивалентности

- Цикл [*инкрементальное увеличение отводимого времени*]
  - Проверить эквивалентность
    - Построить схему-митру
    - Построить КНФ-представление (кодировка Цейтина)
    - Запустить SAT-решатель (DPLL/CDCL) *с ограничениями по времени*
    - Если статус определен: выход из цикла
  - Упростить схему-митру (rewriting)
    - Если статус определен: выход из цикла
  - Редуцировать схему-митру (functional reduction)
    - Если статус определен: выход из цикла
- Если если статус не определен
  - Финальная проверка эквивалентности (*с большими ресурсами*)
- Если статус SAT
  - Выдать контрпример

Функциональная редукция

- рандомизированная симуляция
- определение классов эквивалентности узлов [modulo inputs]
- случайная симуляция, пока не стабилизируются классы
- Пока есть нетривиальные классы [и ограничение по времени]
  - взять пару узлов (n_1, n_2) из одного класса [топологический порядок]
  - построить промежуточную схему-митру и проверить выполнимость
    - если UNSAT ($n_1 \sim n_2$), слить n_1 и n_2 в один узел
    - если достигнуто ограничение по ресурсам, попробовать другую пару
    - иначе построить контрпример α
  - построить множество $A' = \{\alpha' \mid \rho_h(\alpha', \alpha) = 1\}$ [биты вне конуса случайны]
  - симуляция, пока не стабилизируются классы
    - для различающих наборов α' строятся новые (см. выше)

Практикум №2

- Реализуйте на языке программирования C или C++ алгоритм DPLL (базовый или с эвристиками)
  - Без использования рекурсии
  - Только правило распространения единицы
  - Приемлемая производительность и потребление памяти
  - Вход: файл с КНФ в формате DIMACS
  - Выход: вердикт (SAT или UNSAT)

Формат DIMACS [CNF] представления КНФ

- Пример представления КНФ в нотации DIMACS [CNF]

c CNF w/ 4 variables and 3 clauses	
`p cnf 4 3`	
`1 3 -4 0`	$(x_1 \vee x_3 \vee \overline{x}_4)$
`4 0`	x_4
`2 -3`	$(x_2 \vee \overline{x}_3)$

- Тестовые задачи для оценки решателей: https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html