Инструменты символической проверки моделей

Занятие №11

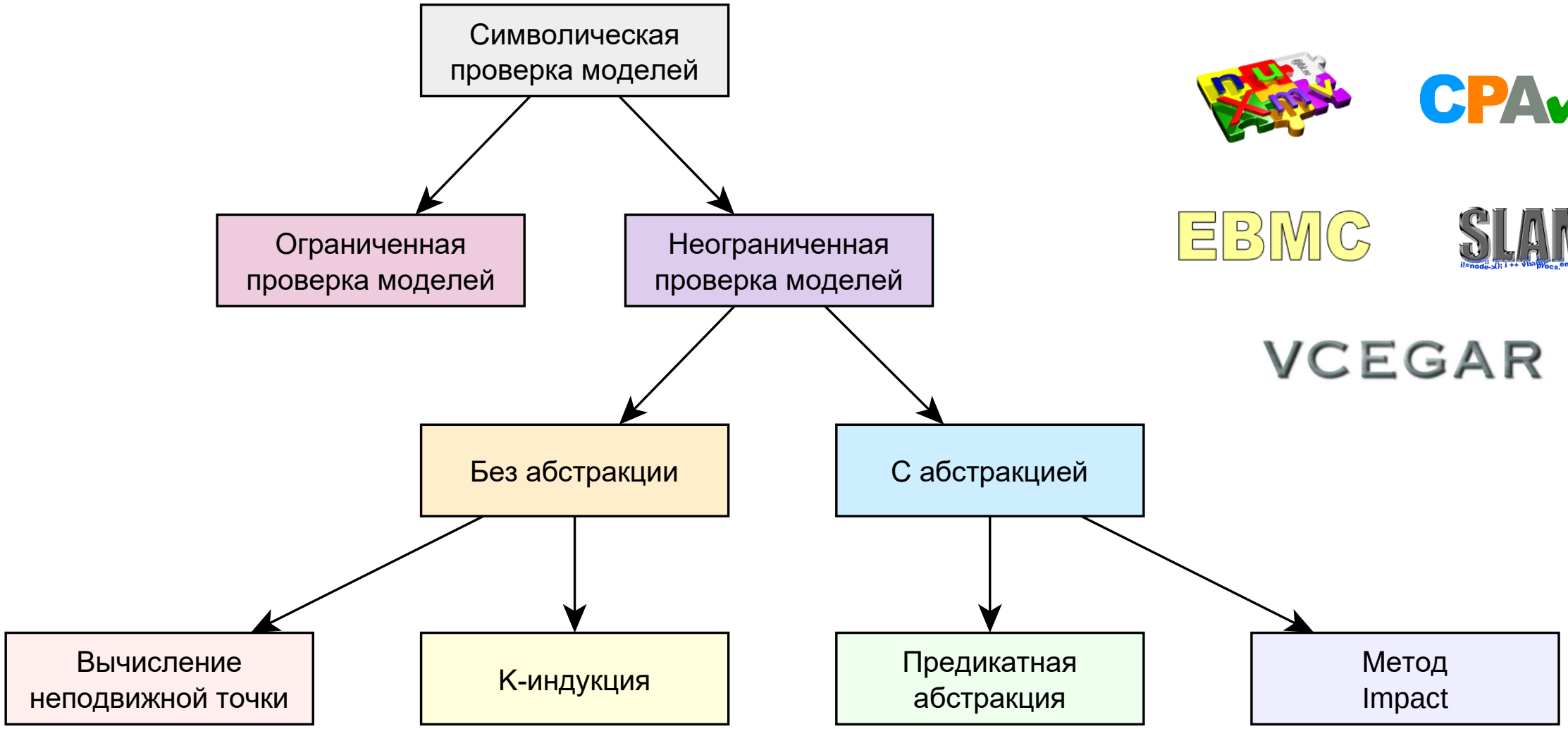
Любая схема представляется в виде системы уравнений, составленных из символов, соответствующим различным реле и переключателям схемы. <…> этот аппарат в точности аналогичен исчислению высказываний символической логики.

К. Шеннон.
Символический анализ релейных и переключательных схем

Александр Сергеевич Камкин

kamkin@ispras.ru

Символическая проверка моделей



Инструменты семейства SMV

- **CMU SMV, Cadence SMV**
  - Кеннет Макмиллан, 1990-ые – 2000-ые
  - Университет Карнеги – Меллона (CMU)

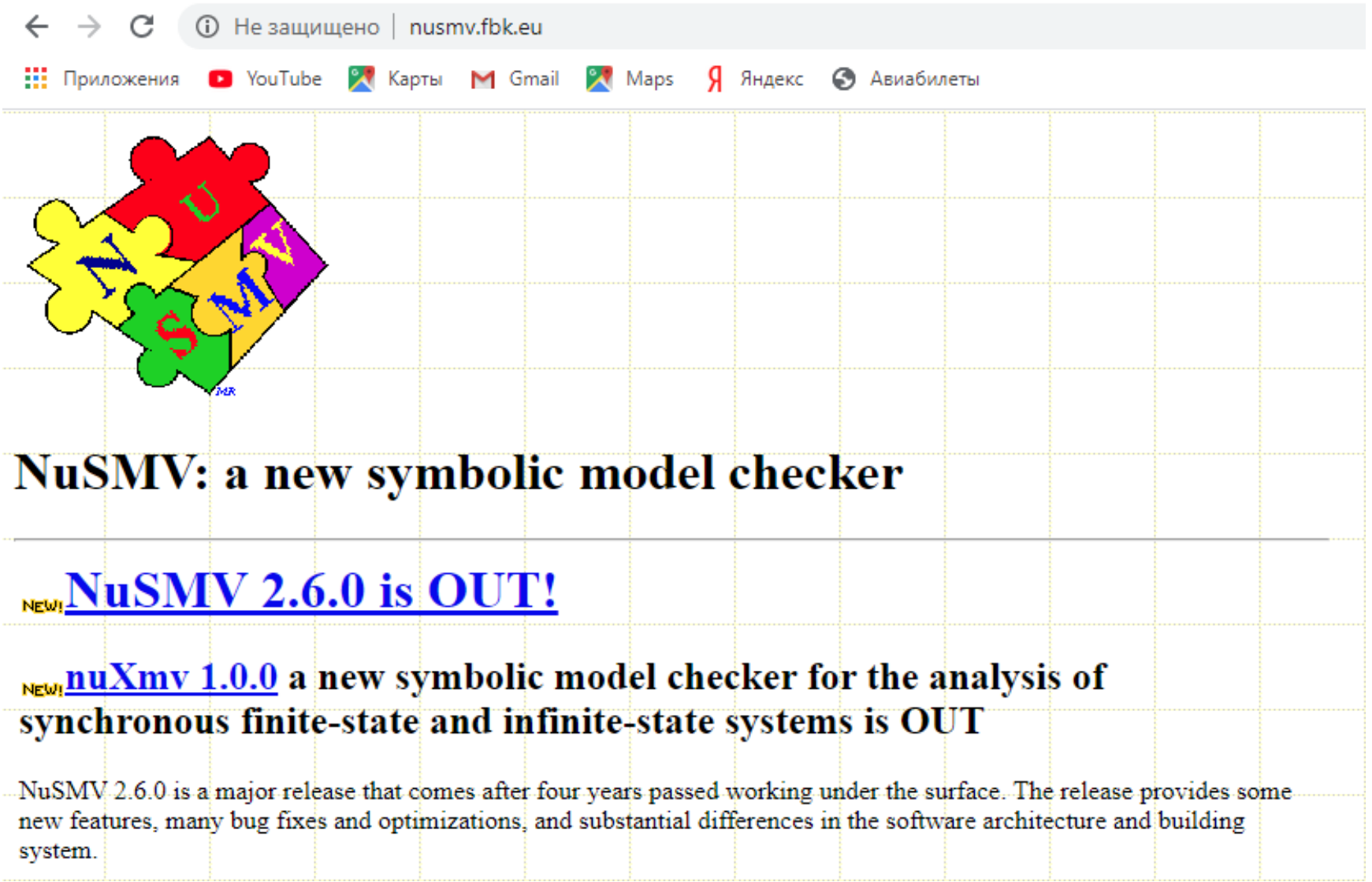
Kenneth L. McMillan

- **NuSMV, nuXmv**
  - Роберто Кавада et. al., 2010-ые
  - Центр Бруно Кесслера (FBK)

Основные возможности инструментов

- **Моделирование параллельных систем**
  - Иерархические модули (в духе Verilog и VHDL)
  - Синхронный параллелизм + недетерминизм
- **Спецификация темпоральных требований**
  - LTL (linear-time Temporal Logic)
  - CTL (Computation Tree Logic)
- **Символическая проверка моделей**
  - BDD-based model checking
  - SAT-based model checking

NuSMV: http://nusmv.fbk.eu/



Пример модели на SMV

```
MODULE main                    /-- точка входа --/
 VAR                            -- переменные
  request : boolean;
  state   : {ready, busy};
 ASSIGN                         -- функция переходов
  init(state) := ready;
  next(state) :=
   case
    state = ready & request = TRUE : busy
    TRUE : {ready, busy}
   esac
```

Базовые типы данных

Тип данных	Множество значений	Битовая длина
boolean	$\{\text{FALSE}, \text{TRUE}\}$	1
integer	$[\text{INT\_MIN} + 1, \text{INT\_MAX}]$	зависит от реализации, обычно 32
word$[n]$ **unsigned word**$[n]$	$[0, 2^n - 1]$	$n \geq 1$
signed word$[n]$	$[-2^{n-1}, 2^{n-1} - 1]$	$n \geq 1$

Литералы базовых типов данных

- **boolean**: `TRUE`, `FALSE`
- **integer**: `0`, `1`, `-1`, `2`, `-2`, … (только десятичные)
- **word**: `0` $[\ signed\ ]\ base\ [\ width\ ]\ \_\ value$
  - `0sb5_10111` -- *знаковое слово длины 5*
  - `0uo6_37` -- *беззнаковое слово длины 6*
  - `0d11_9` -- *беззнаковое слово длины 11*
  - `0sh12_a9` -- *знаковое слово длины 12*

Другие типы данных

- **Перечисление (enumeration)**
  - Символическое: `{stopped, running, waiting}`
  - Целочисленное: `{-1, 0, 1}`
  - Смешанное: `{-1, YES, NO}`
- **Массив (array)**
  - **`array`** `0..15` **`of boolean`**
  - **`array`** `-1..1` **`of`** `{stopped, running, waiting}`
  - **`array`** `1..10` **`of array`** `1..2` **`of integer`**

Объявление переменных

```
-- булева переменная
flag   : boolean;
-- целочисленная переменная
index  : integer;
-- беззнаковое слово (битовый вектор) длины 1
turn   : word[1];
-- определение статуса процесса
status : {stopped, running, waiting};
-- массив из 16 флагов
flags  : array 0..15 of boolean;
```

Операции в порядке приоритета

Операции	Ассоциативность	Комментарий
`[]  [ : ]`	слева направо	Индексирование массива, выбор битов слова
`!`	справа налево	Логическое и побитовое отрицание
`::`	слева направо	Конкатенация слов
`-`	справа налево	Унарный минус
`*  /  mod`	слева направо	Умножение, деление, остаток от деления
`+  -`	слева направо	Сложение, вычитание
`<<  >>`	слева направо	Сдвиг влево, сдвиг вправо
`union`	слева направо	Объединение множеств
`in`	слева направо	Принадлежность элемента множеству
`=  !=  <  <=  >  >=`	слева направо	Сравнение равно / не равно и меньше / больше
`&`	слева направо	Логическое и побитовое И
`|  xor  xnor`	слева направо	ИЛИ, исключающее ИЛИ, исключающее ИЛИ-НЕ
`? :`	слева направо	Условная операция
`<->`	слева направо	Логическая и побитовая эквивалентность
`->`	справа налево	Логическая и побитовая импликация

Некоторые особенности операций

- нет неявного приведения между **boolean**, **integer**, **signed** / **unsigned word**$[n]$
- бинарные операции над словами (кроме сдвига и конкатенации) требуют, чтобы операнды имели одинаковую знаковость и разрядность
- побитовые операции не применимы к целым числам
- результатом операций выбора битов и конкатенации всегда является беззнаковое слово

Выбор битов и конкатенация

```
0sb8_10101100[4:1]     -- 0ub4_0110
0ub8_10101100[4:1]     -- результат тот же
0sb4_1010::0sb4_1100   -- 0ub8_1010_1100
0sb4_1010::0ub4_1100   -- результат тот же
0ub4_1010::0sb4_1100   -- результат тот же
0ub4_1010::0ub4_1100   -- результат тот же
```

Встроенные функции (1)

Функция	Описание
$\mathbf{abs}(t)$	абсолютная величина t
$\mathbf{min}(t_1, t_2)$	минимум из двух величин t_1 и t_2
$\mathbf{max}(t_1, t_2)$	максимум из двух величин t_1 и t_2
$\mathbf{count}(b_1, \dots, b_n)$	число истинных логических выражений из $b_1, \dots, b_n$
$\mathbf{sizeof}(w)$	длина слова w (целое число)
$\mathbf{word1}(b)$	приведение значения истинности к типу $\mathbf{unsigned\ word}[1]$
$\mathbf{bool}(t)$	приведение 1-битного беззнакового слова или целого числа t к типу **boolean**
$\mathbf{toint}(t)$	приведение константного слова (знакового или беззнакового) или значения истинности t к типу **integer**
$\mathbf{uwconst}(C, n)$	приведение целочисленной константы C к типу $\mathbf{unsigned\ word}[n]$
$\mathbf{swconst}(C, n)$	приведение целочисленной константы C к типу $\mathbf{signed\ word}[n]$
$\mathbf{unsigned}(w)$	преобразование слова w в беззнаковое слово той же длины
$\mathbf{signed}(w)$	преобразование слова w в знаковое слово той же длины

Встроенные функции (2)

Функция	Описание
$\mathbf{extend}(w, \Delta n)$	расширение слова w (Δn — число дополнительных бит): • если слово беззнаковое, добавляются нули; • иначе — дублируется знаковый бит
$\mathbf{resize}(w, n)$	изменение длины слова w (n — новая длина): • если $n = m$, где $m = \mathbf{sizeof}(w)$, возвращается слово w; • если $n > m$ — $\mathbf{extend}(w, n - m)$; • если $n < m$: ○ если слово беззнаковое — $w[n-1:0]$; ○ иначе — $\mathbf{signed}(w[m-1:m-1] :: w[n-2:0])$;

Примеры приведения типов

```
word1(FALSE)            -- 0ub1_0
word1(TRUE)             -- 0ub1_1
bool(0ub1_0)            -- FALSE
bool(0ub1_1)            -- TRUE
bool(0)                 -- FALSE
bool(123)               -- TRUE (как отличное от нуля целое)
toint(FALSE)            -- 0
toint(TRUE)             -- 1
toint(0uh4_F)           -- 15
toint(0sh4_F)           -- -1
uwconst(15, 4)          -- 0ub4_1111
swconst(-1, 4)          -- 0sb4_1111
unsigned(0sh4_F)        -- 0uh4_F (слово становится беззнаковым)
signed(0uh4_F)          -- 0sh4_F (слово становится знаковым)
extend(0ub4_1010, 1)    -- 0ub5_0_1010 (добавляется нулевой бит)
extend(0sb4_1010, 1)    -- 0sb5_1_1010 (дублируется знаковый бит)
resize(0ub4_1010, 3)    -- 0ub3_010 (обрезается старший бит)
resize(0sb4_1010, 3)    -- 0sb3_110 (переносится знаковый бит)
```

Модули и их экземпляры (1)

```
MODULE M(p_1, ..., p_n) -- имя и параметры
 VAR
  x_1 : T_1; ...          -- переменные и подмодули
 ASSIGN
  init(x_1) := t_1; ...   -- начальные значения
  next(x_1) := s_1; ...   -- значения после перехода
 DEFINE
   y_1 := f_1; ...        -- связки (макросы)
```

Модули и их экземпляры (2)

```
MODULE M(p_1, ..., p_n) -- имя и параметры
 VAR
  x_1 : T_1; ...    -- переменные и подмодули
 INIT
  φ(x_1, ...);      -- начальное состояние
 INVAR
  ψ(x_1, ...);      -- инвариант состояния
 TRANS
  δ(x_1, next(x_1), ...); -- отношение переходов
```

Пример: 3-битный счетчик (1)

```
MODULE counter_cell(carry_in)
  VAR
    value : boolean;
  ASSIGN
    init(value) := FALSE;
    next(value) := value xor carry_in;
  DEFINE
    carry_out := value & carry_in;
```

Пример: 3-битный счетчик (2)

```
MODULE main
  VAR
    bit0 : counter_cell(TRUE);
    bit1 : counter_cell(bit0.carry_out);
    bit2 : counter_cell(bit1.carry_out);
```

Средства спецификации требований

```
MODULE M(...)
   ...
   JUSTICE простая-формула   -- FAIRNESS
   ...
   JUSTICE простая-формула   -- GF(...)

   LTLSPEC формула-LTL
   ...
   LTLSPEC формула-LTL
```

Пример: алгоритм Петерсона

```
while true do
   NCS_i: /* некритическая секция */
   SET_i: flag_i := 1; turn := i;
   TST_i: while (flag_1-i = 1) ∧ (turn = i) do
             skip /* ожидание */
          end;
   CRS_i: /* критическая секция */
   RST_i: flag_i := 0
end
```

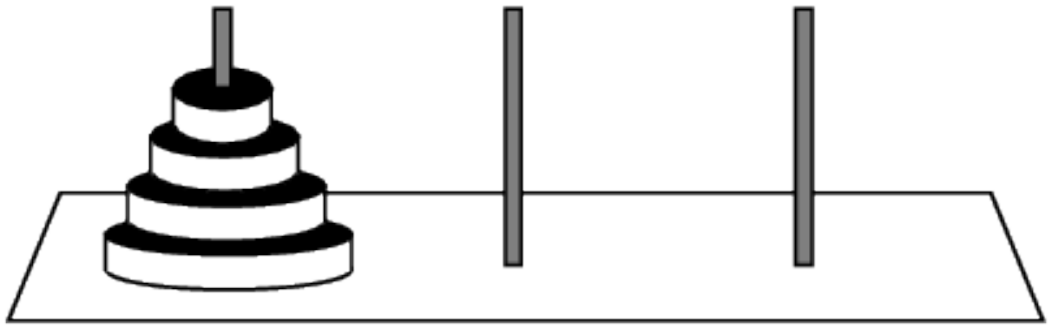
Решение головоломок (напоминание)

Найти **последовательность шагов** для достижения некоторой **цели**

$$\neg \mathbf{F}\{goal\}$$

$$\downarrow$$

$$\text{Контрпример} = \text{Решение}$$



http://mathworld.wolfram.com

Пример: волк, коза и капуста

- Крестьянину нужно перевезти через реку волка, козу и капусту
- В лодке может поместиться крестьянин, а с ним либо волк, либо коза, либо капуста
- Если оставить волка с козой на берегу, волк съест козу
- Если оставить козу с капустой, коза съест капусту
- Только когда крестьянин рядом, коза и капуста в безопасности

https://illuminations.nctm.org

Как крестьянину перевезти свой груз?

Домашнее задание

- С помощью инструмента NuSMV (или nuXmv) решите головоломку «Ханойская башня»