# Применение раскрашенных сетей Петри для верификации спецификаций коммуникацтонных протоколов

Стененко А. А.

Научный руководитель Непомнящий В. А.

# Введение

- Верификация распределённых систем, то есть проверка того что такая система обладает данными свойствами, является актуальной проблемой современного программирования.

- При верификации как правило используются модели систем. Для моделирования могут использоваться сети Петри и конечные автоматы.

- Цель работы — создание системы верификации автоматных спецификаций и раскрашенных сетей Петри.

# Автоматные спецификации

- Система взаимодействующих расширенных конечных автоматов (РКА) состоит из
  - автоматов
  - глобальных переменных
  - ожидающих приёма сигналов окружения Env.

- Содержит инициализирующую функцию, которая задаёт начальные значения переменных и начальные сигналы окружения Env.

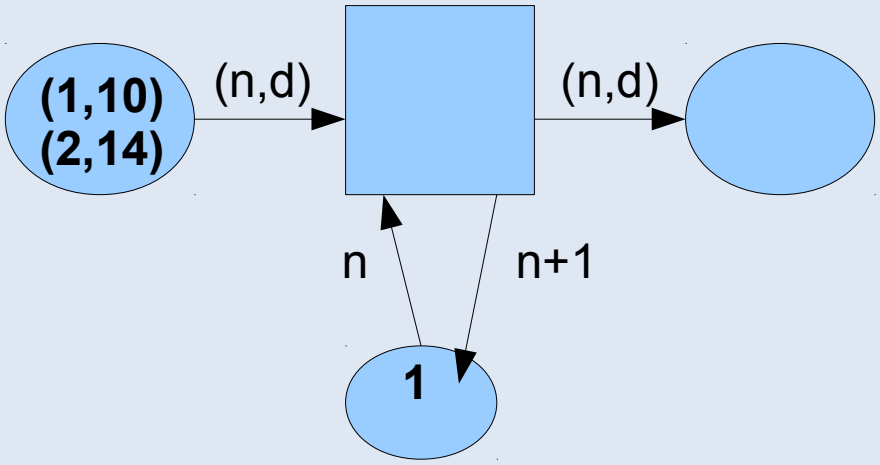
# Автоматные спецификации

- РКА имеет состояния и переходы.
- Оперирует локальными и глобальными переменными
- Взаимодействует со средой: при осуществлении переходов принимает и отправляет сообщения следующего вида:

  Кортеж (Src, Dst, Type, Param)

# Язык раскрашенных сетей Петри.

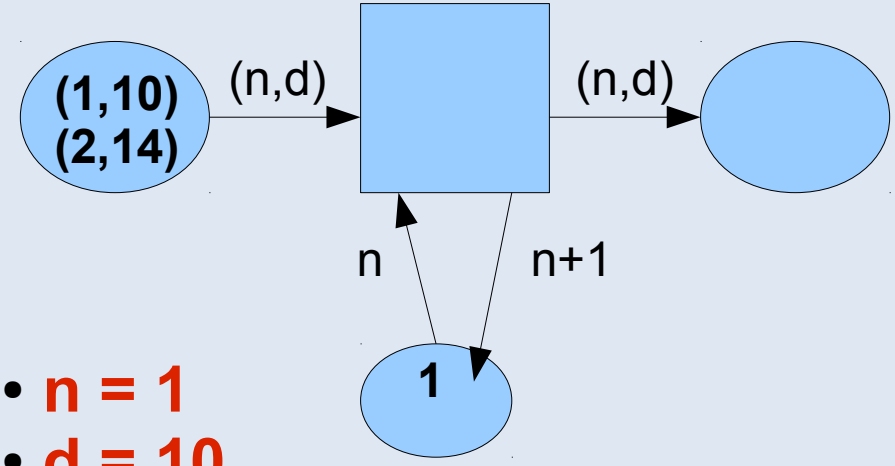
- РСП — двудольный орграф, вершины которого — места и переходы.
- Места могут содержать метки, являющиеся значениями различных типов данных.
- Переходы осуществляются в соответствии с выражениями приписанными дугам сети (используется язык основанный на SML).




Типы данных:
- INT
- BOOL
- UNIT
- Перечисления
- Кортежи
- Списки

# Язык раскрашенных сетей Петри.

- РСП — двудольный орграф, вершины которого — места и переходы.
- Места могут содержать метки, являющиеся значениями различных типов данных.
- Переходы осуществляются в соответствии с выражениями приписанными дугам сети (используется язык основанный на SML).



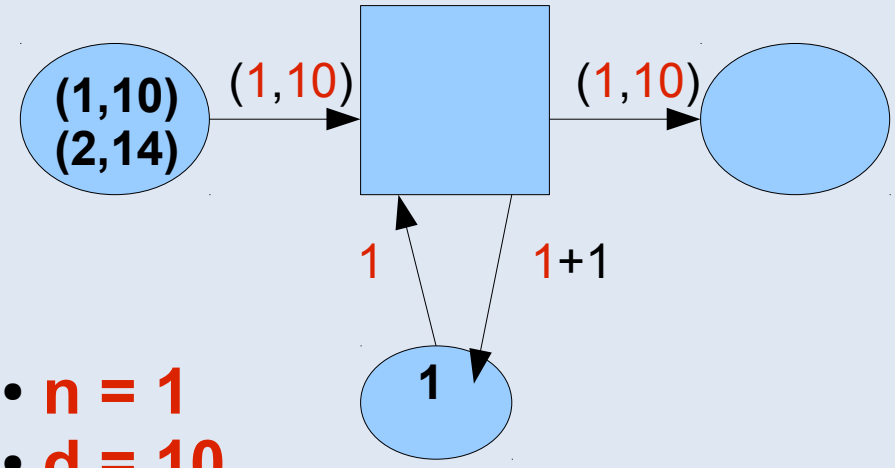

- **n = 1**
- **d = 10**

Типы данных:

- INT
- BOOL
- UNIT
- Перечисления
- Кортежи
- Списки

# Язык раскрашенных сетей Петри.

- РСП — двудольный орграф, вершины которого — места и переходы.
- Места могут содержать метки, являющиеся значениями различных типов данных.
- Переходы осуществляются в соответствии с выражениями приписанными дугам сети (используется язык основанный на SML).

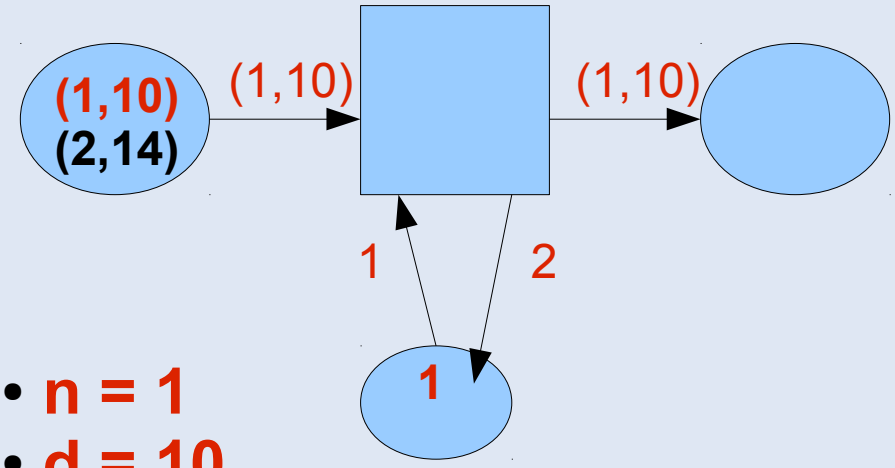



- **n = 1**
- **d = 10**

Типы данных:
- INT
- BOOL
- UNIT
- Перечисления
- Кортежи
- Списки

# Язык раскрашенных сетей Петри.

- РСП — двудольный орграф, вершины которого — места и переходы.
- Места могут содержать метки, являющиеся значениями различных типов данных.
- Переходы осуществляются в соответствии с выражениями приписанными дугам сети (используется язык основанный на SML).

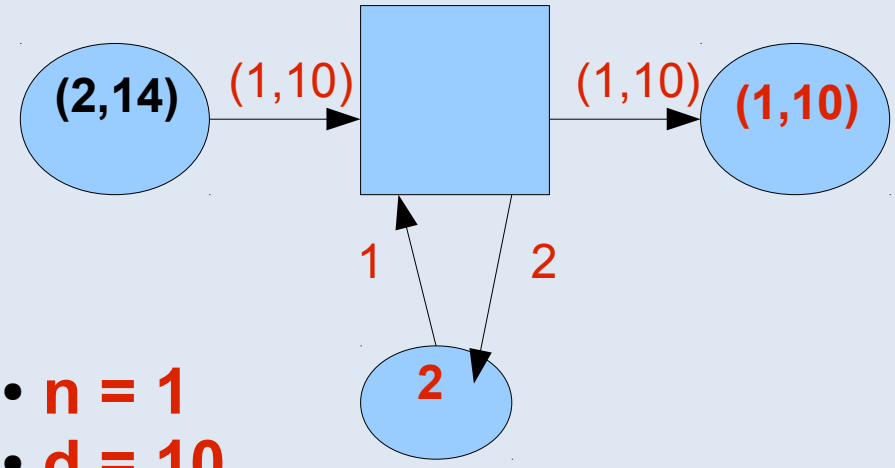



- **n = 1**
- **d = 10**

Типы данных:
- INT
- BOOL
- UNIT
- Перечисления
- Кортежи
- Списки

# Язык раскрашенных сетей Петри.

- РСП — двудольный орграф, вершины которого — места и переходы.
- Места могут содержать метки, являющиеся значениями различных типов данных.
- Переходы осуществляются в соответствии с выражениями приписанными дугам сети (используется язык основанный на SML).



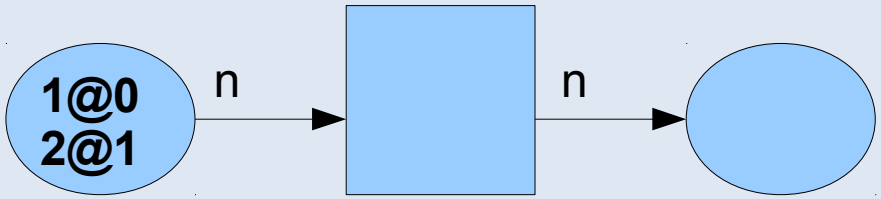

- **n = 1**
- **d = 10**

Типы данных:
- INT
- BOOL
- UNIT
- Перечисления
- Кортежи
- Списки

# Время в РСП

- Для каждой фишки кроме её цвета определён момент времени, начиная с которого она доступна
- Есть глобальный счётчик времени, определяющий текущий момент

T=0

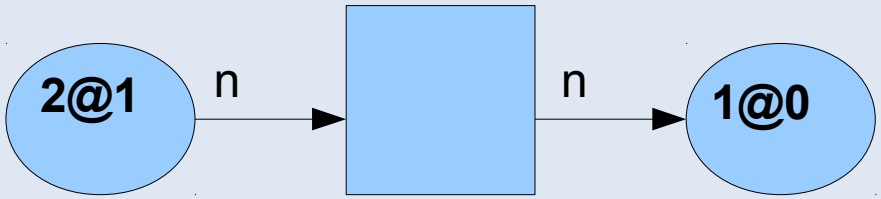



Сначала выполнится переход с n = 1

# Время в РСП

- Для каждой фишки кроме её цвета определён момент времени, начиная с которого она доступна
- Есть глобальный счётчик времени, определяющий текущий момент

T=1




Переход с n = 2 выполнится после

# РСП: ограничения

- Установлена верхняя граница ёмкости мест и длины списков

- Язык выражений включает в себя:
  - арифметико-логические операторы
  - операторы работы с кортежами, записями и списками
  - условный оператор
  - операторы времени
  - определяемые пользователем функции (используется C)

# РСП: ограничения

Ограничение на время:

- время относительное, нет глобального таймера «с начала исполнения»: выражения на дугах, имеют вид «expr» (без времени), либо «expr @+ Δt»
  - Оператор «@+» добавляет время к текущему моменту.
- Оператор «@», указывающий абсолютный момент времени, может использоваться только в выражениях начальной разметки.

# Язык Promela

- Является базовым языком системы верификации SPIN
- Позволяет делать вставки на языке C.

**Пример**

```
#define MAX 10
int p1, p2, p3, p4;

active proctype net() {
    atomic { p1 = 1; p2 = 2; p4 = 1; }
loop:
    atomic {
        if
        :: p1 > 0 && p2 < MAX && p3 < MAX ->
            p1--; p2++; p3++;
            goto loop;
        :: p2 > 0 && p3 > 0 && p1 < MAX && p4 < MAX ->
            p2--; p3--; p1++; p4++;
            goto loop;
        :: else -> skip;
        fi;
    }
}
```

# Представление РСП

- Для представления РСП в системе CPN Tools используется XML:

```
<place id="ID14">
  <text>nr</text>
  <type>
    <text>INT</text>
  </type>
  <initmark>
    <text>1`1++1`2</text>
  </initmark>
</place>
```

# Верификация РСП

- Для верификации РСП-моделей используются:
  - Транслятор из РСП в язык Promela
  - Система верификации SPIN

- SPIN позволяет проверять свойства описанные с помощью LTL, например, такие как:
  - «Когда-то в будущем выполнится данный предикат»
  - «Данный предикат будет истинным всегда, начиная с текущего состояния»

# Этапы трансляции

- Чтение модели, построение внутреннего представления.
- Поочерёдно транслируются
  - Типы данных
  - Переменные
  - Места
  - Переходы

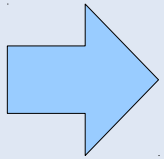
# Трансляция типов данных Сетей Петри

- Типы данных ML транслируются в типы данных C

```
colset INT = int;
```

```
typedef int cs_int;
```

```
colset pair = product int * int;
```



```
typedef struct cs_pair {
        cs_int field1;
        cs_int field2;
} cs_pair;
```

# Трансляция типов-списков

```
colset INT_L = list INT;
```

```
typedef struct cs_INT_L {
        inttype length;
        cs_INT
        content[LIST_MAX_LENGTH];
} cs_INT_L;
```

# Трансляция типов со временем

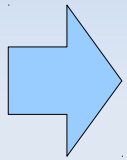
```
colset T_INT = INT timed;
```

```
typedef struct cs_T_INT {
        intype time;
        cs_INT value;
} cs_T_INT;
```

# Трансляция мест сетей Петри

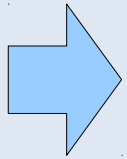
- Разметка места, имеющего тип T, представляется переменной языка C, имеющей тип «мулитимножество над T»

Мультимножество над int



```
typedef struct msOfint {
        int size;
        cs_int content[MS_MAX_SIZE];
} msOfint;
```

Мультимножество над bool



```
typedef struct ms1Ofbool {
        int count[2];
} ms1Ofbool;
```

# Трансляция мест сетей Петри

- Для каждого из типов определим функции сравнения, присвоения нулевого значения
- Необходимо учитывать не только явно определённые типы, но и все типы, встречающиеся в выражениях

# Трансляция переходов сетей Петри

- Для каждого перехода генерируются следующие фрагменты кода:
  - проверка допустимости перехода
  - осуществление перехода

- Ключевым моментом трансляции переходов является трансляция выражений языка ML, приписанных дугам сети Петри:
  - В процессе обхода дерева разбора выражение транслируется в последовательность трёхадресных инструкций на C
  - Для хранения промежуточных результатов используются служебные переменные
  - Проверка допустимости значений аргументов и результата

# Как устроена транслированная модель

- Инициализация

  Главный цикл:

- Поиск значений переменных перехода, делающих его допустимым; если нашли такие значения, то выбираем недетерминированно:
  - либо продолжить поиск (при этом запомнив что в текущем состоянии есть допустимые переходы)
  - либо приступить к выполнению перехода, затем перейти на метку «Главный цикл»
- Если вышли из цикла то состояние будем считать принадлежащим пространству состояний только если не было найдено допустимых переходов.

# Моделирование времени

- Храним относительное время
- Если не было найдено допустимых переходов, уменьшаем время:
  - Находим фишку с минимальным ненулевым временем, и вычитаем это время

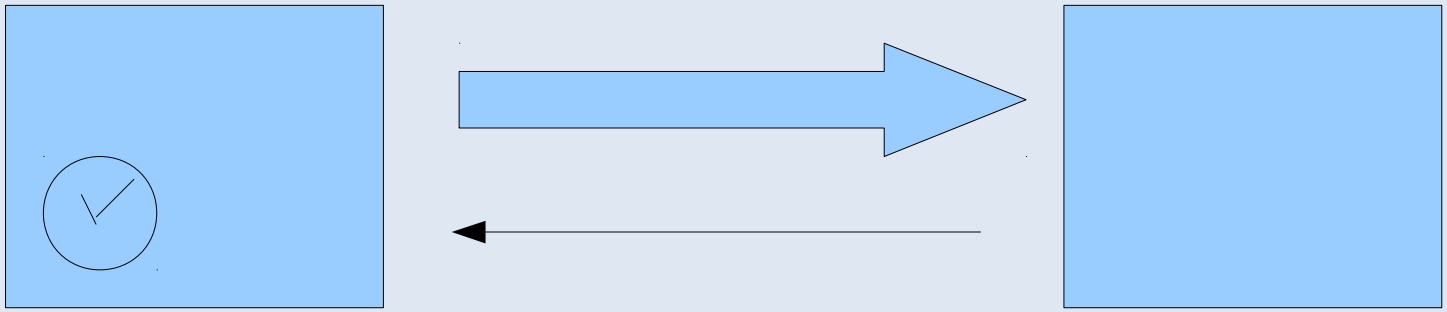
# Эффективность транслированной модели

- Группирование инструкций с помощью Promela-конструкции «`atomic`», исключающей промежуточные состояния из пространства состояний модели.
- Использование вставок на языке C, скрывающих служебные переменные и позволяющих писать более простой код, чем код на Promela.
- Оптимизация представления значений мультимножеств над разными типами данных с учётом количества возможных значений.
- Возможно исключение части пространства состояний модели из рассмотрения с помощью определения пользовательских предикатов.

# Пример: протокол «PAR»

[Таненбаум, Компьтерные сети]

- Отправитель и получатель в ненадёжной среде
- Подтверждение доставки
- Таймер для повторной отправки



# Пример: протокол «PAR»

- 57 мест, 35 переходов
- ~ 9000 строк на Promela

# Пример: протокол «Par»

- Свойство: всегда последовательность принятых пакетов является префиксом последовательности пакетов, которые отправляются

- Время работы верификатора 0.06 с

# Вывод верификатора

```
Full statespace search for:
never claim         + (never_0)
assertion violations    + (if within scope of claim)
acceptance   cycles    - (not selected)
invalid end states - (disabled by never claim)
State-vector 1508 byte, depth reached 7491, errors: 0
     1110 states, stored
      226 states, matched
     1336 transitions (= stored+matched)
    23741 atomic steps
hash conflicts:         0 (resolved)
Stats on memory usage (in Megabytes):
    1.626 equivalent memory usage for states (stored*(State-vector + overhead))
    2.348 actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534 memory used for DFS stack (-m10000)
    4.401 other (proc and chan stacks)
  135.401       total actual memory usage
```

# Верификация телефонных сетей с дополнительными функциональностями

Построена автоматная модель телефонной сети, смоделированы функциональности:

переадресация звонков, если номер занят, CFB

безусловная переадресация, CFU

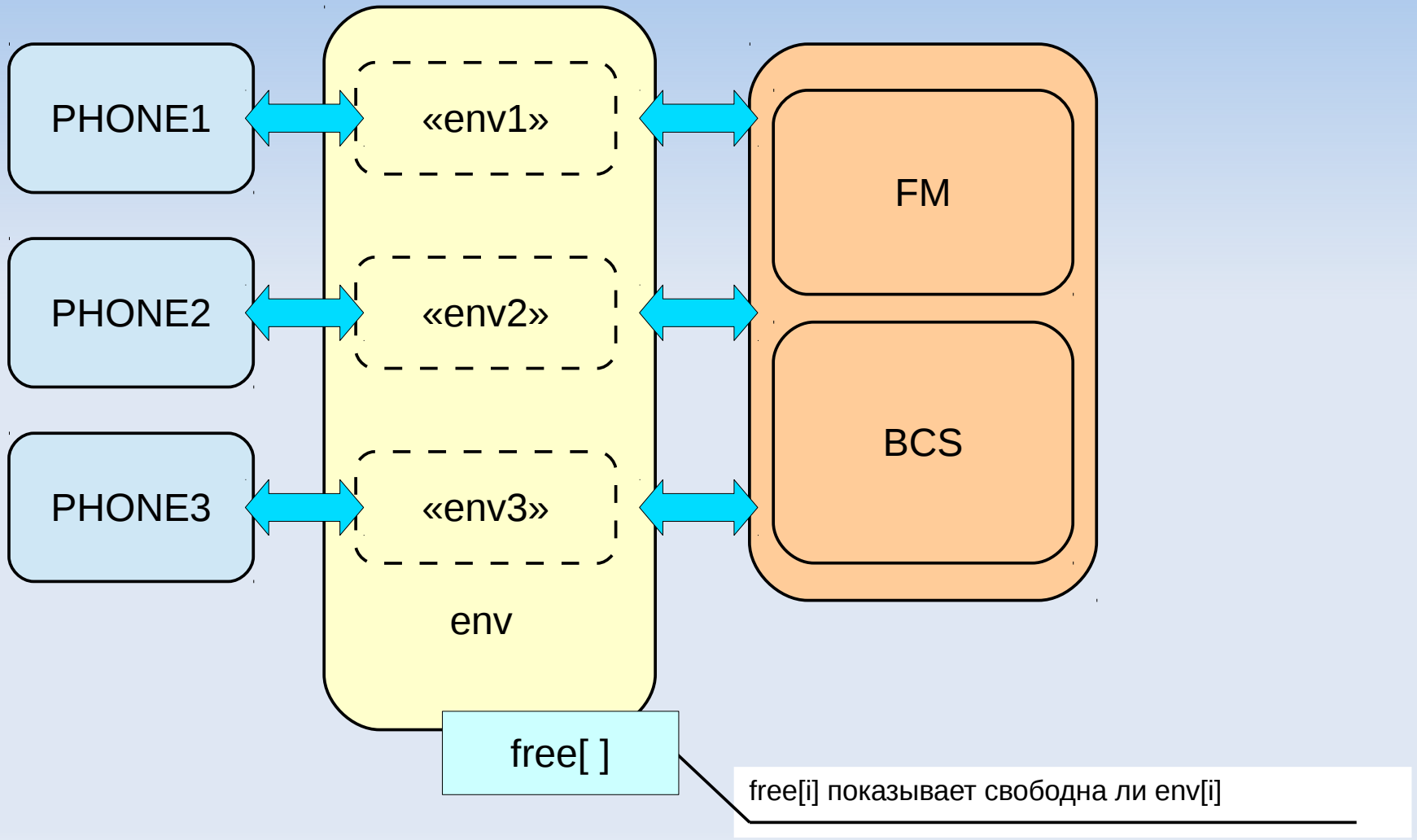
прямое соединение, DC

запрет исходящих звонков на заданный номер, ODS

запрет входящих звонков с заданного номера, TCS

запрет всех входящих звонков, DT

# Верификация телефонных сетей с дополнительными функциональностями



# Верификация телефонных сетей с дополнительными функциональностями

- Сгенерированы модели с четырьмя телефонами и двумя функциональностями.

- Проверялись свойства:
  - Отсутствие тупиков
  - Свойство безопасности:

    Всегда выполнен предикат (для разных функциональностей разные предикаты).

    Например, если подключена безусловная переадресация, то абоненту не позвонят.
  - Отсутствие зацикливаний обработчика функциональностей.

# Верификация телефонных сетей с дополнительными функциональностями

- Третье свойство было нарушено если в модели присутствуют две переадресации.

- Модель была исправлена и верификация произведена повторно;

  без ошибок.

- Размер модели:
  - ~ 200 000 состояний;
  - ~ 800 000 переходов.
- Потребление памяти 4 гб (+использование диска).
- Время работы верификатора модели порядка минуты.

# Заключение

- Реализована система верификации автоматных спецификаций и раскрашенных сетей Петри.
- Проведены эксперименты по верификации моделей телекоммуникационных систем, представленных в виде систем взаимодействующих автоматов и РСП.

# Спасибо за внимание!