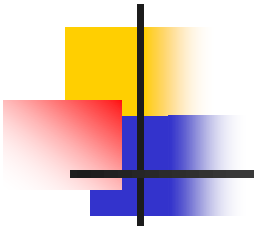
Верификация параллельных программных и аппаратных систем

Курс лекций

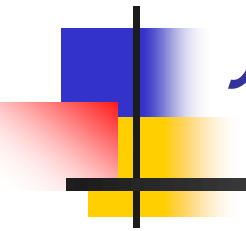
Шошмина Ирина Владимировна

Карпов Юрий Глебович



План курса

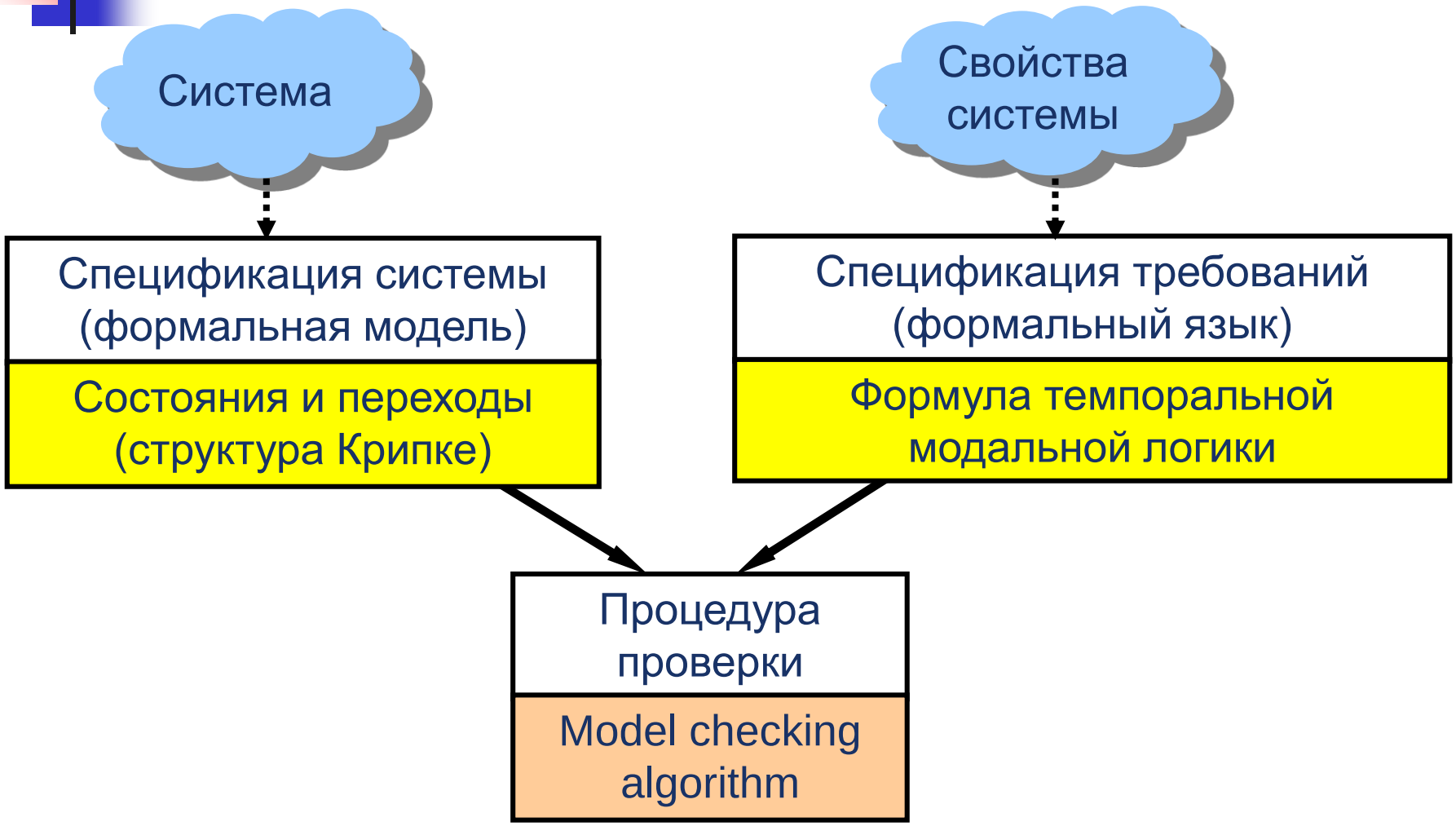
1. Введение
2. Метод Флойда-Хоара доказательства корректности программ
3. Темпоральные логики
4. Автоматный подход к проверке выполнения формул LTL
5. Система верификации Spin и язык Promela. Примеры верификации
6. Алгоритм Model checking для проверки выполнимости формул CTL
7. BDD и их применение
8. Символьная проверка моделей
9. Структура Крипке как модель реагирующих систем
10. Применения метода верификации model checking
11. Темпоральные свойства систем
12. Количественный анализ дискретных систем при их верификации
13. Верификация систем реального времени
14. Консультации по курсовой работе



Лекция 7

Алгоритм Model checking для проверки выполнимости формул CTL

Задача верификации методом Model checking



Наша задача – рассмотреть алгоритм Model checking для логики **CTL**

Структура Крипке = множество возможных миров + бинарное отношение на нем + разметка

$M = \langle AP, W, L, R \rangle$

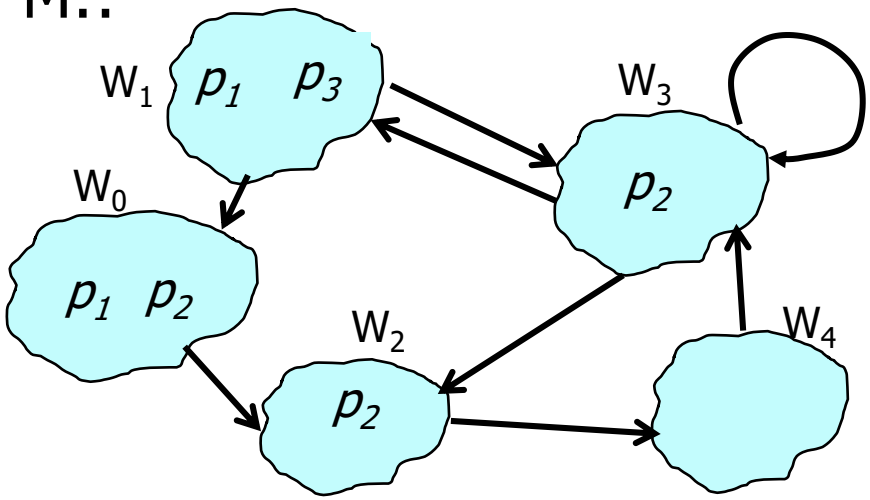
$AP = \{p_1, \ldots, p_n\}$ множество атомов

$W = \{w_0, \ldots\}$ множество "миров"

$L: W \to 2^{AP}$ функция разметки

$R \subseteq W \times W$ отношение достижимости

M::



"*Возможные миры*" дают явное описание возможных вариантов состояния дел или возможного хода событий (*т.е. их можно использовать для представления динамики систем*).

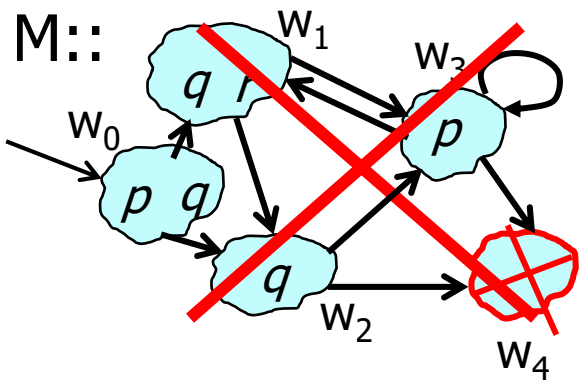
В конкретных приложениях отношение R имеет конкретный смысл.

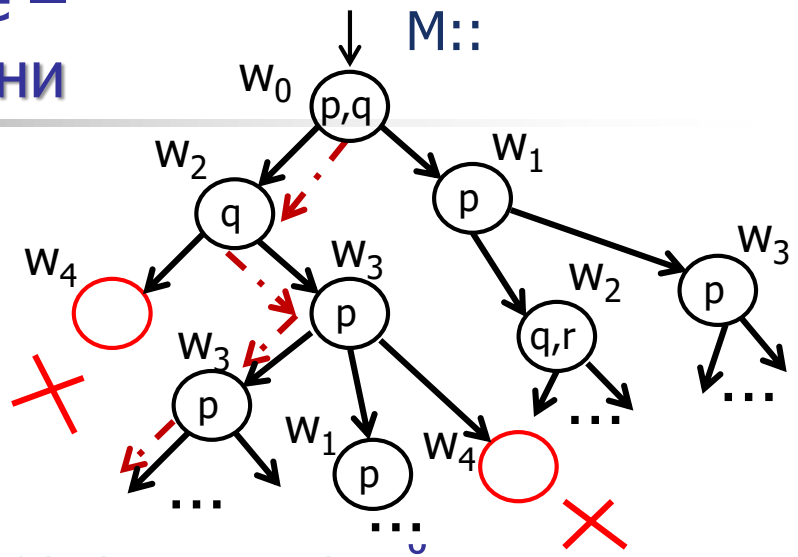
В дискретных системах (модели схем и программ): ***ВОЗМОЖНЫЕ МИРЫ*** – состояния системы в различные дискретные моменты времени,

АТОМАРНЫЕ ПРЕДИКАТЫ – интересующие нас атомарные утверждения,

ОТНОШЕНИЕ ДОСТИЖИМОСТИ R – переходы между состояниями.

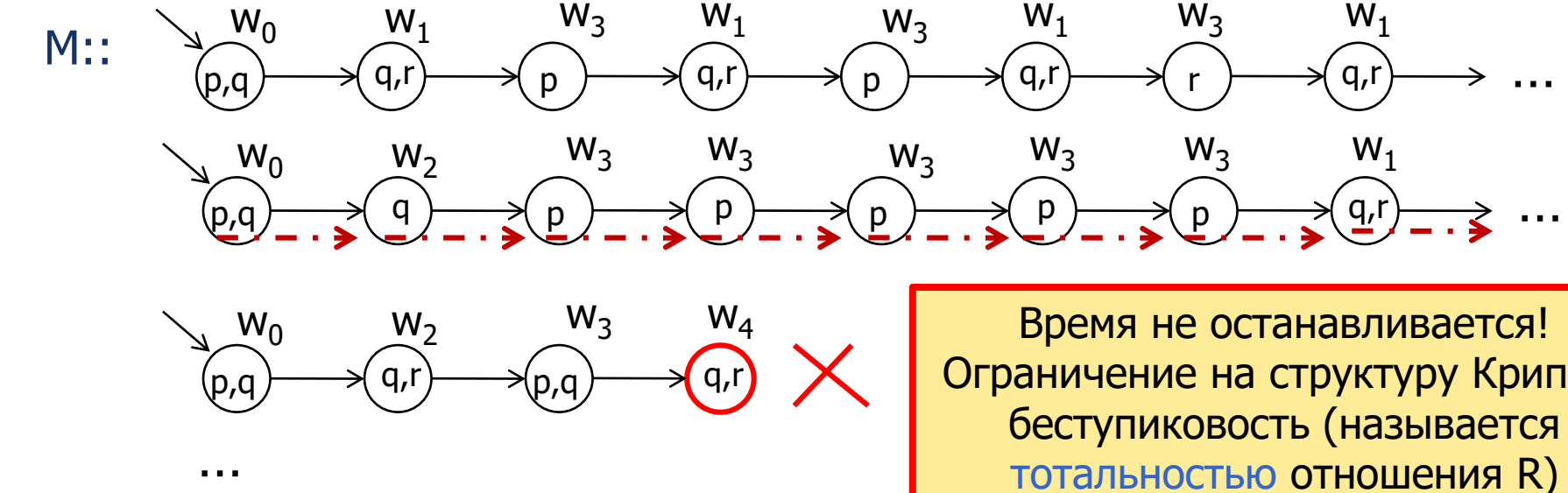
Развертки структур Крипке – динамика систем во времени





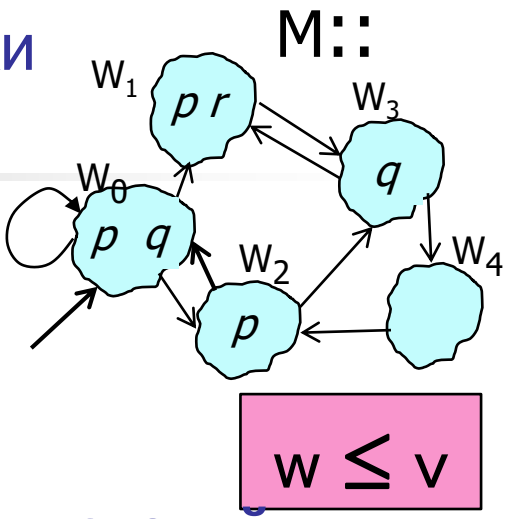
дерево вычислений

множество вычислений



Время не останавливается!
Ограничение на структуру Крипке: беступиковость (называется тотальностью отношения R)

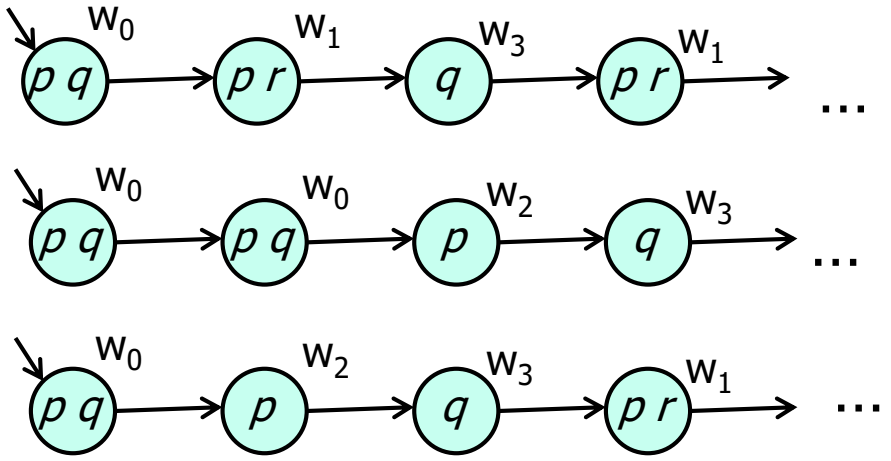
Структуры Крипке и модальные логики линейного и ветвящегося времени



- Для формул темпоральных логик определяем СЕМАНТИКУ на РАЗВЕРТКАХ моделей Крипке с отношением R – ПОЛНЫМ или ЧАСТИЧНЫМ порядком МОМЕНТОВ времени.

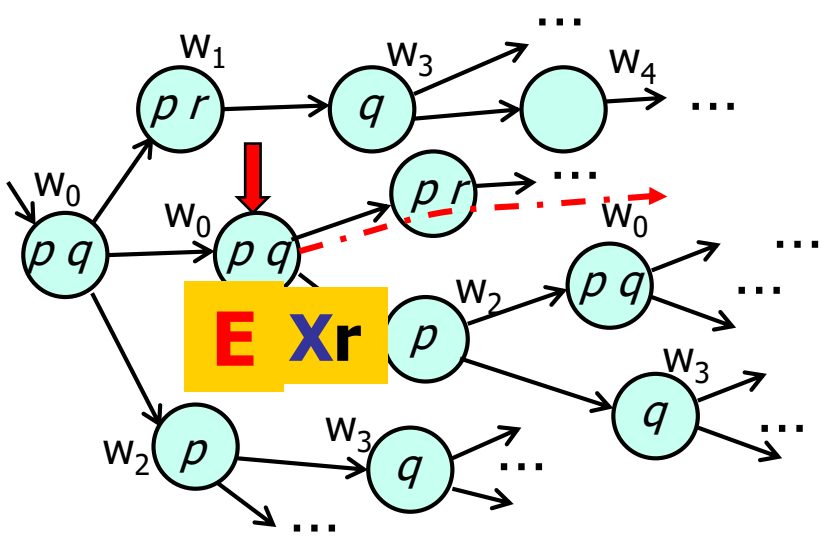
множество вычислений

R – полный порядок



дерево вычислений

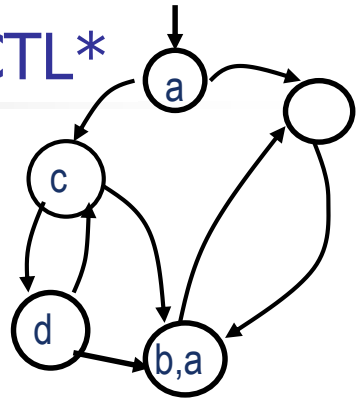
R – частичный порядок



Логики линейного времени

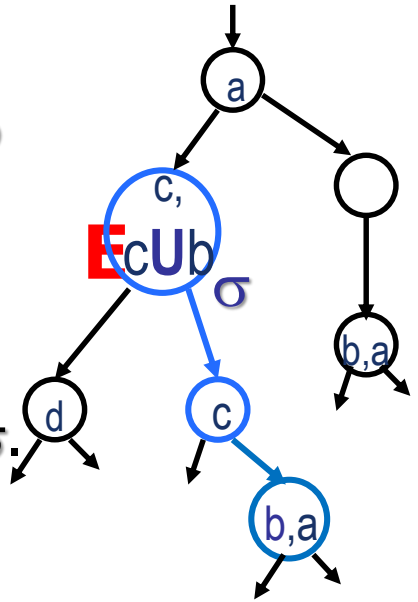
Логики ветвящегося времени

Расширенная логика ветвящегося времени CTL*



Темпоральные логики ветвящегося времени рассматривают деревья вычислений (ветвящиеся пути) - на развертке структуры Крипке.

CTL* – Computational Tree Logic* - это одна из возможных логик ветвящегося времени.



Грамматика. Формула CTL* - это формула состояний φ

- *формулы состояний* $\varphi ::= p \mid \neg\varphi \mid \varphi\vee\varphi \mid E\psi$
- *формулы путей* $\psi ::= \varphi \mid \neg\psi \mid \psi\vee\psi \mid \psi U\psi \mid X\psi$

если s является начальным состоянием пути σ, то формулу φ состояния S можно считать формулой пути σ.

A ψ, G ψ, F ψ - выводимые формулы

Формула пути имеет смысл только если зафиксирован путь!

В состояниях могут стоять только формулы состояний!

LTL и CTL – подклассы CTL*

Формулы LTL:

AG(p ⇒**F** q)

A (¬a∨ **G**b & (a**U** ¬c))

A (a **U** ¬b)

Формулы CTL:

AG(p&¬**E**F(q⇒r))

EF(a & **E**(a**U** ¬c))

A (a **U** ¬ b)

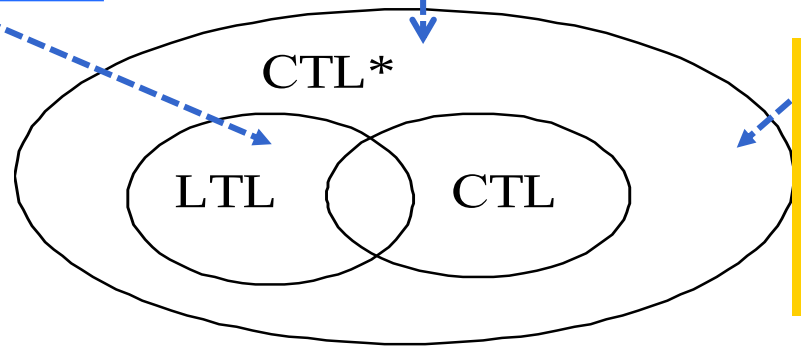
Формулы CTL*:

E(¬p & **X** **A** **F** q)

EG (p ⇒**F** q)

A (a **U** ¬ (**F** b))

Рассматриваются все пути из начального состояния



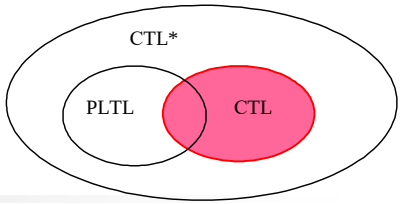
Не только формула, но и свойство **E**G(p ⇒**F** q) не выразимы в CTL и LTL

Алгоритм проверки выполнимости формул CTL* очень сложен. На практике используются подклассы этой темпоральной логики – LTL и CTL.

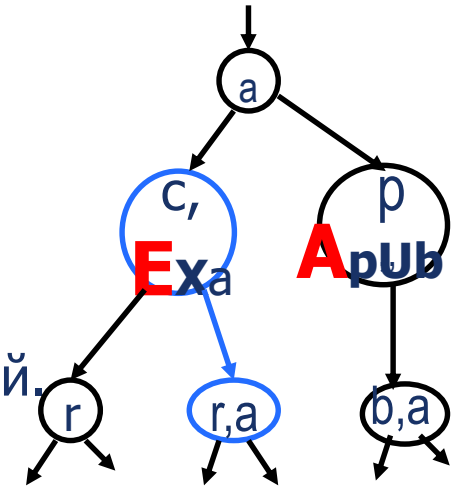
LTL - формулы пути. При верификации считаем, что они должны выполняться для всех вычислений, (в CTL*) они предваряются квантором пути A.

В CTL каждый темпоральный оператор предваряется квантором пути A или E.

CTL – синтаксис



В **CTL** каждый темпоральный оператор предварен квантором пути. Из-за этого
ВСЕ ФОРМУЛЫ CTL – формулы СОСТОЯНИЙ



4 темпоральных оператора: **X, F, G, U**

2 квантора пути: **E** – существует путь, **A** – для всех путей.

Всего 8 базовых CTL-операторов:

AX и **EX**, **AG** и **EG**,
AF и **EF**, **AU** и **EU**

Возможные формулы: **E**[c**U**b], **A**[p**U**(r∨**EF**q)], **EX**p∧**EX**q, **EGAF** p, ...

Нельзя в CTL выразить **EGF**p, **A** [**X**p ∨ **XX**r], ...

CTL – ограниченная логика по сравнению с CTL*,
но она чрезвычайно удобна из-за эффективности алгоритма проверки выполнимости ее формул на структурах Крипке

Неформальное определение семантики CTL

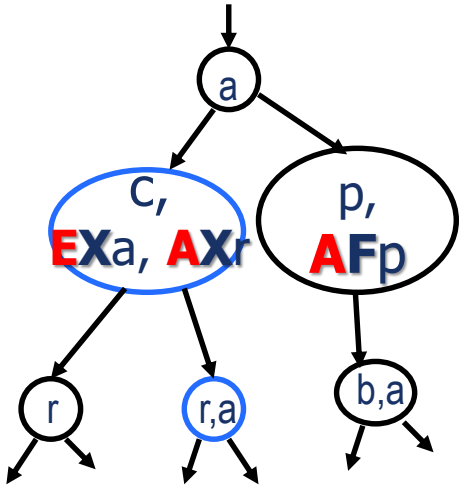
Синтаксис (грамматика):

$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{AX}\varphi \mid \mathbf{EX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{AG}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{A}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{E}[\varphi_1 \mathbf{U} \varphi_2]$

Все формулы CTL – это формулы состояний!!

AXφ – формула φ выполняется во всех *следующих* состояниях (непосредственных потомках текущего состояния)

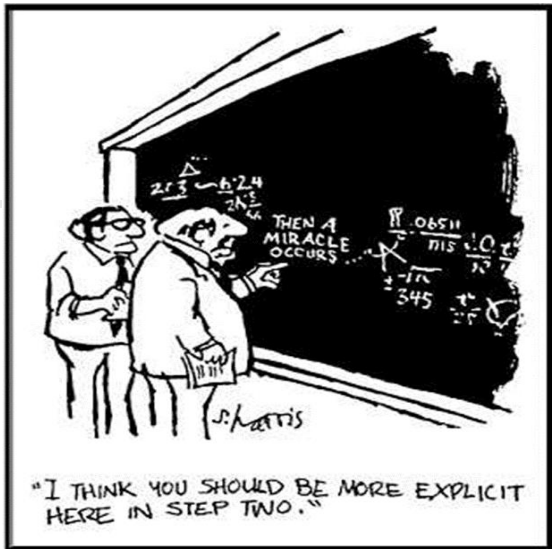
EXφ - формула φ выполняется хотя бы в одном *следующем* состоянии



AFφ (*неизбежно* φ) - на всех путях из текущего состояния формула φ *когда-нибудь* выполнится

EFφ (*возможно* φ) - из текущего состояния существует путь, на котором формула φ *когда-нибудь* выполнится

Формальная семантика CTL задается на структуре Крипке M



$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{A}\psi \mid \mathbf{E}\psi$

$\psi ::= \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2$

Path(s) – все вычисления, начинающиеся в s

$\pi = s_0 s_1 s_2 s_3 \ldots$ - бесконечное вычисление

s_j – j-ое состояние вычисления π

$M,s \models p \equiv p \in L(s)$

$M,s \models \neg\varphi \equiv s \not\models \varphi$

$M,s \models \varphi_1 \vee \varphi_2 \equiv s \models \varphi_1 \vee s \models \varphi_2$

$M,s \models \mathbf{A}\psi \equiv (\forall \pi: \pi \in Paths(s))\ M,\pi \models \psi$

$M,s \models \mathbf{E}\psi \equiv (\exists \pi: \pi \in Paths(s))\ M,\pi \models \psi$

$M,\pi \models \mathbf{X}\varphi \equiv M,s_1 \models \varphi;$

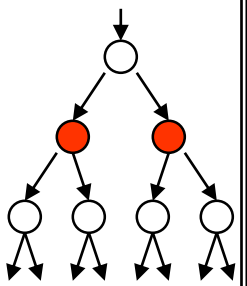
$M,\pi \models \varphi_1 \mathbf{U} \varphi_2 \equiv (\exists j: 0 \le j)\ (M,s_j \models \varphi_2 \wedge (\forall k: 0 \le k < j)\ M,s_k \models \varphi_1)$

Формула CTL выполняется на структуре Крипке $M \models \varphi$ ттт она выполняется во всех начальных состояниях этой структуры

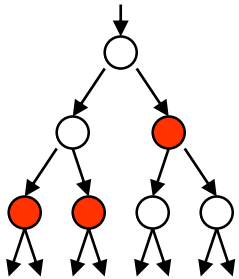
CTL – рекурсивное задание

$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U \varphi] \mid E[\varphi U \varphi]$

AXred:

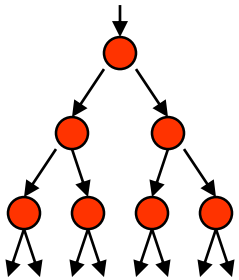


AFred:



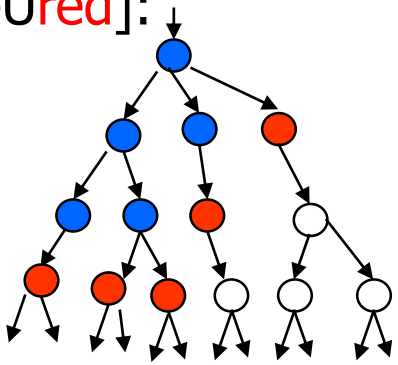
$AF\varphi = \varphi \vee AX\ AF\varphi$

AGred:



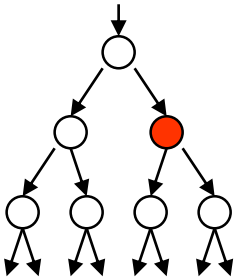
$AG\varphi = \varphi \wedge AX\ AG\varphi$

A [blueUred]:

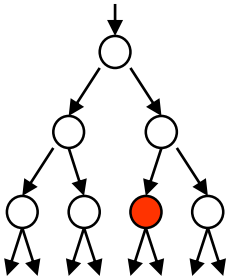


$A\ [\varphi U \phi] = \phi \vee \varphi \wedge AX\ A[\varphi U \phi]$

EXred:

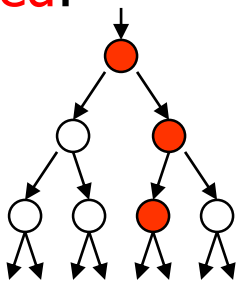


EFred:



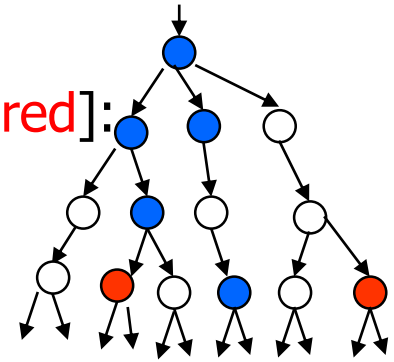
$EF\varphi = \varphi \vee EX\ EF\varphi$

EGred:



$EG\varphi = \varphi \wedge EX\ EG\varphi$

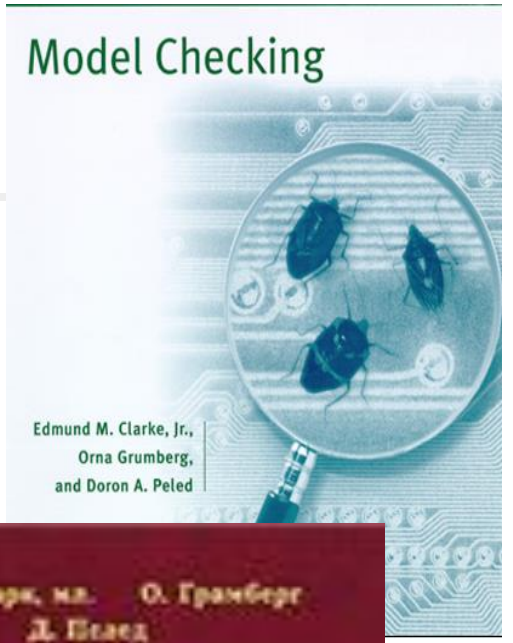
E [blueUred]:

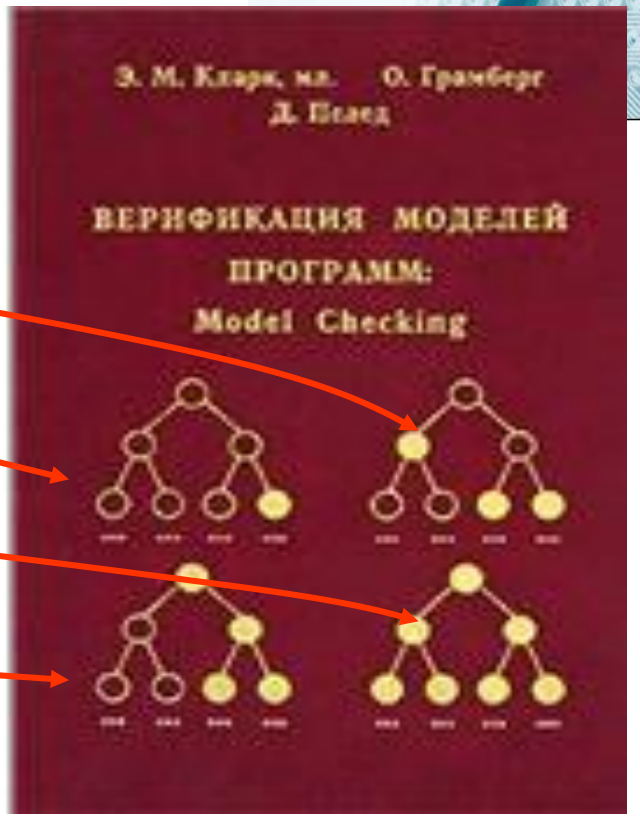


$E[\varphi U \phi] = \phi \vee \varphi \wedge EX\ E[\varphi U \phi]$

Литература

- E. M.Clarke, O. Grumberg, D.Peled. Model checking. MIT Press, 1999
  - (Русский перевод: Э.М.Кларк, О.Грамберг, Д.Пелед. Верификация моделей программ: Model Checking. М.,2002)





CTL формулы:

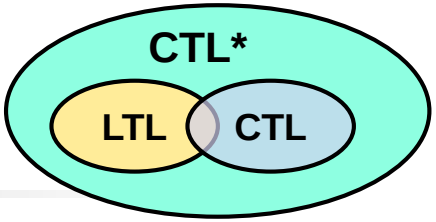
AF φ

EF φ

AG φ

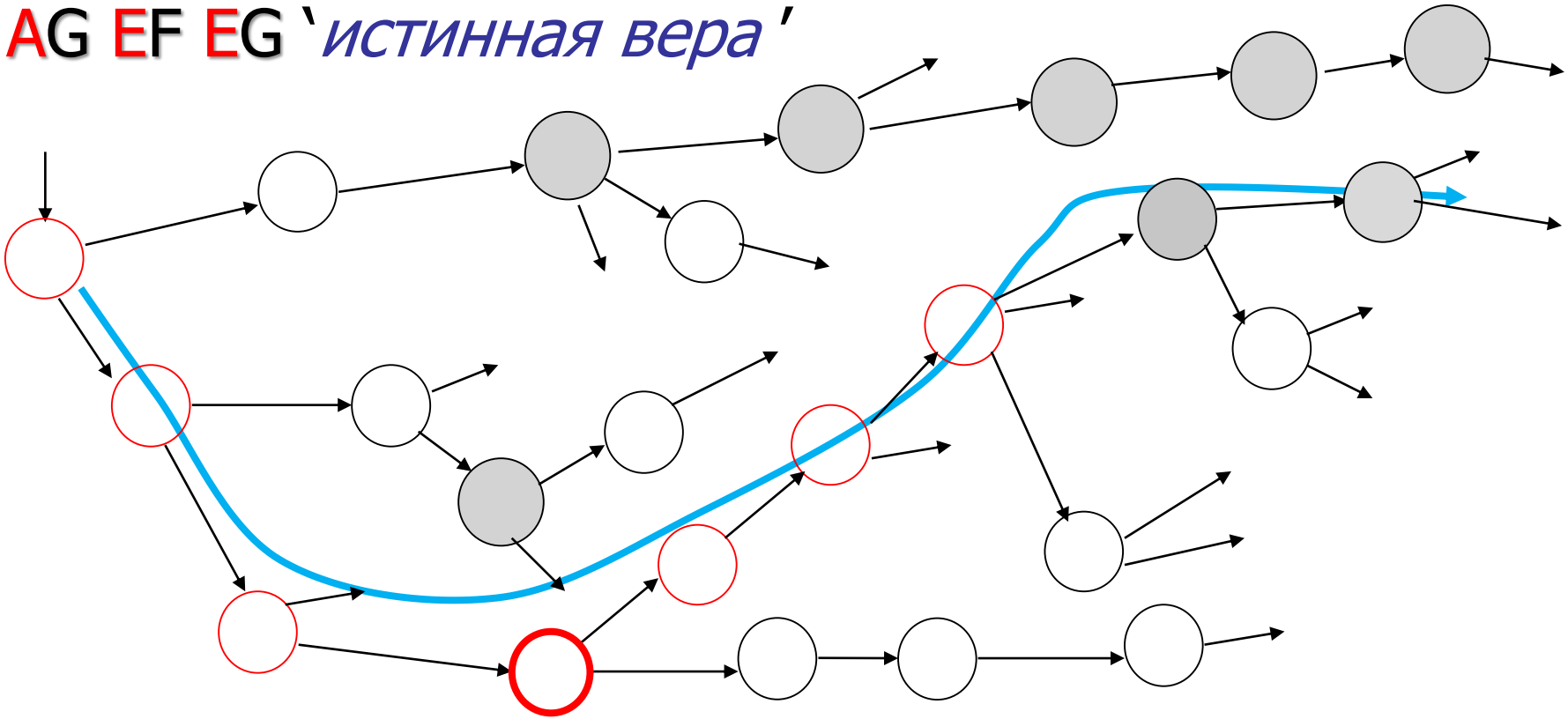
EG φ

Пример свойства, выраженного в CTL



- *'Любой грешник всегда имеет шанс вернуться на путь истинной веры':*

AG EF EG '*истинная вера*'

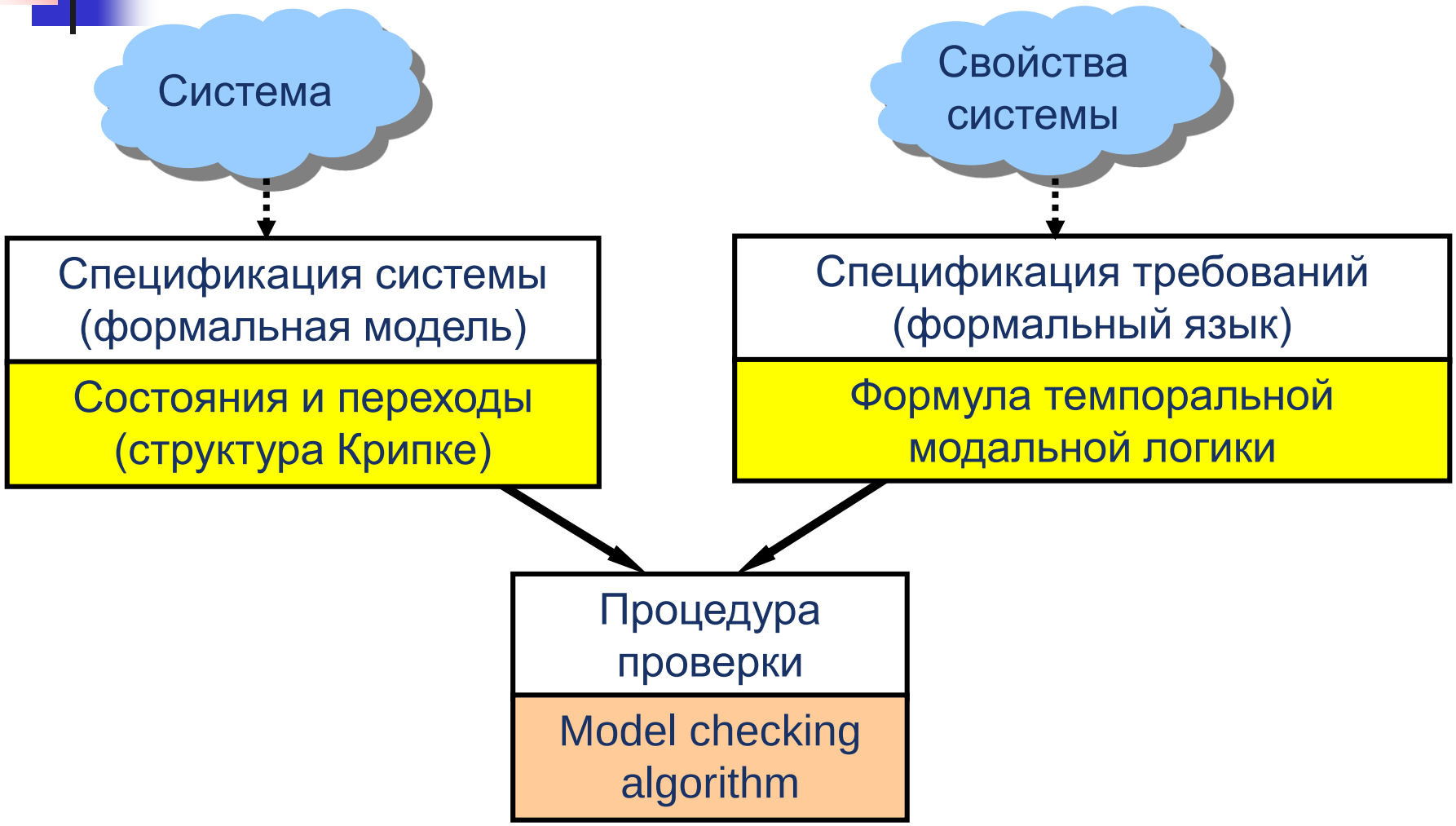


В любом состоянии вашей жизни (сколько бы мы ни грешили AG) существует такой путь (E), что на нем в конце концов (F) попадем в состояние, с которого существует "истинный путь" (EG)

Примеры спецификации требований в CTL

- AGAF Restart
  - *на всех путях, из любого состояния этих путей обязательно вернемся в состояние рестарта*
- AFAG (x<y)
  - *на всех путях когда-то в будущем будет достигнут такой режим, в котором соотношение x<y выполняется постоянно*
- ¬EF(int >0.01)
  - *не существует такого режима работы, при котором интенсивность облучения пациента превысит 0.01 рентген в сек.*
- AG (req ⇒ A (req U ack))
  - *во всех режимах после того, как запрос req установится, он никогда не будет снят, пока на него не придет подтверждение*
- E[p U A [q U r]]
  - *существует режим, в котором условие p будет истинным с начала вычисления до такого состояния, с которого на всех путях q станет непрерывно активным (истинным) до выполнения условия r*

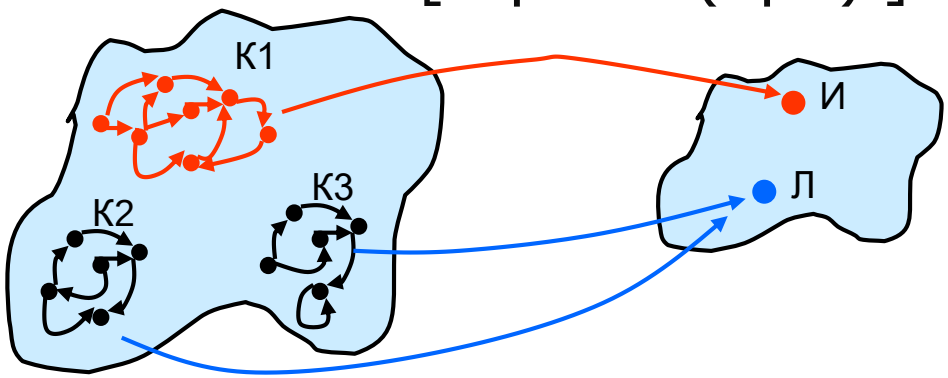
Задача верификации методом Model checking



Наша задача – рассмотреть алгоритм Model checking для логики **CTL**

Проблема Model checking для CTL формул

$\Phi = AG\,[\neg\, p \Rightarrow E(\, q U r)\,]$

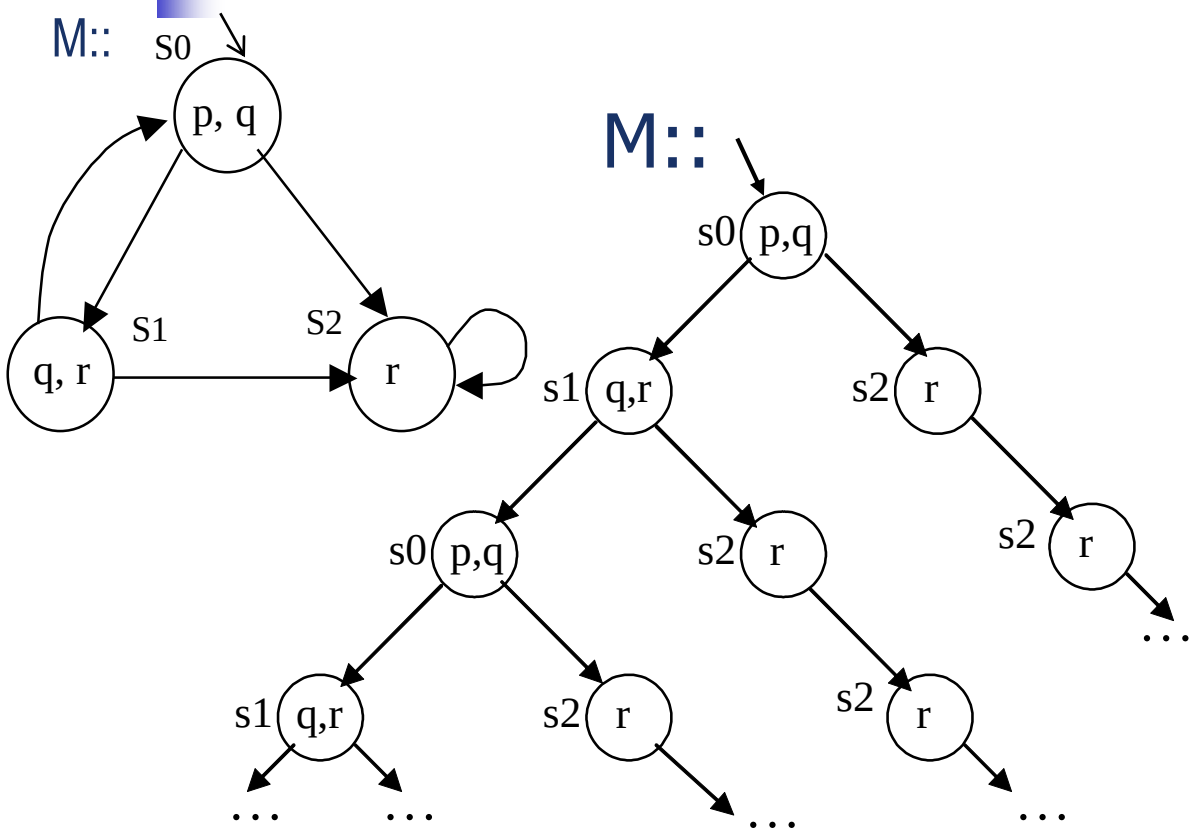


На одной структуре Крипке заданная формула Ф истинна (выполняется), на другой – ложна (не выполняется)

Как проверить выполнимость заданной формулы темпоральной логики ветвящегося времени CTL на произвольной структуре Крипке?

Разработаны элегантные алгоритмы, позволяющие для **ЛЮБОЙ** формулы логики CTL проверить ее выполнимость на **ЛЮБОЙ** заданной структуре Крипке

Model Checking для CTL – проверка на развертке (неформально)



1. M, s_0|= p∧q
2. M, s_0|= **EX** (q∧r)
3. M, s_0|= ¬**AX**(q∧r)
4. M, s_0|= ¬**EF**(p∧r)
5. M, s_0|= ¬**EG**r
6. M, s_0|= **AF**r
7. M,s_0|=**E**[(p∧q)**U**r]
8. M, s_0|= **A**[p **U** r]
9. M, s_0|= **EF AG**r

Этот анализ на бесконечных деревьях вычислений неформален

Необходимо разработать алгоритм проверки выполнимости произвольной темпоральной формулы на любой структуре Крипке.

Композициональность языка формул логики CTL

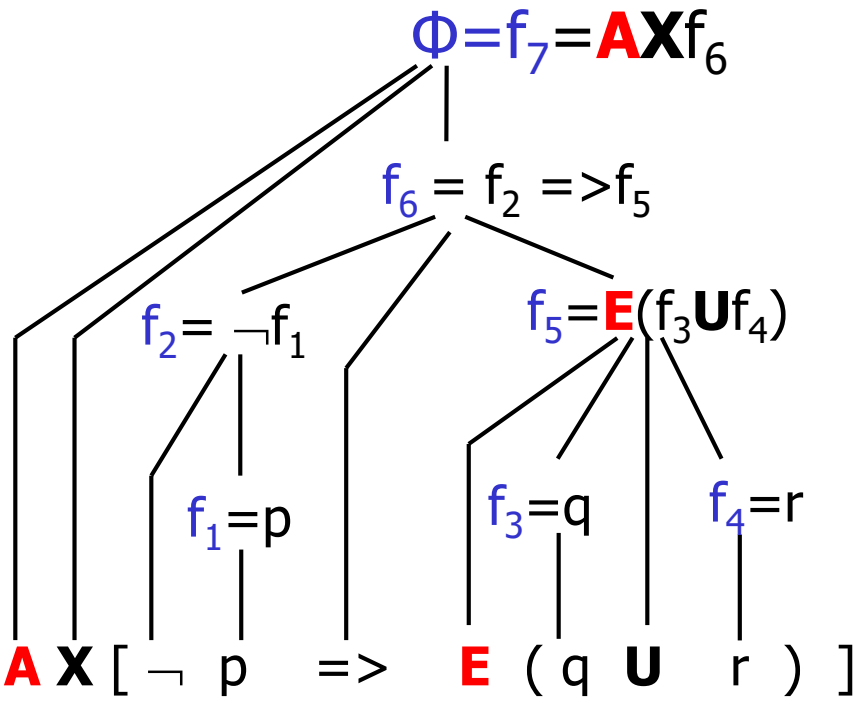
- Язык формул логики CTL является композициональным.

В композициональном языке значение конструкции языка является функцией значений ее частей.

В композициональном языке семантика определяется на основе простейших зависимостей семантических атрибутов: все эти атрибуты являются ***синтезированными***.

Иными словами, семантические атрибуты любой конструкции языка (*нетерминала левой части правила грамматики*) являются функциями семантических атрибутов составляющих конструкцию элементов (*терминалов и нетерминалов правой части правила*.)

A X [¬ p => **E** (q **U** r)]



Синтаксическое дерево формулы CTL

Алгоритм *маркировки* для CTL формул

Формул CTL – бесконечное число !! Как проверить для данной структуры Крипке выполнимость произвольной формулы, например

$$\Phi = \mathbf{AX}\ [\neg p \Rightarrow \mathbf{E}\ (q \mathbf{U} r)\ ]$$

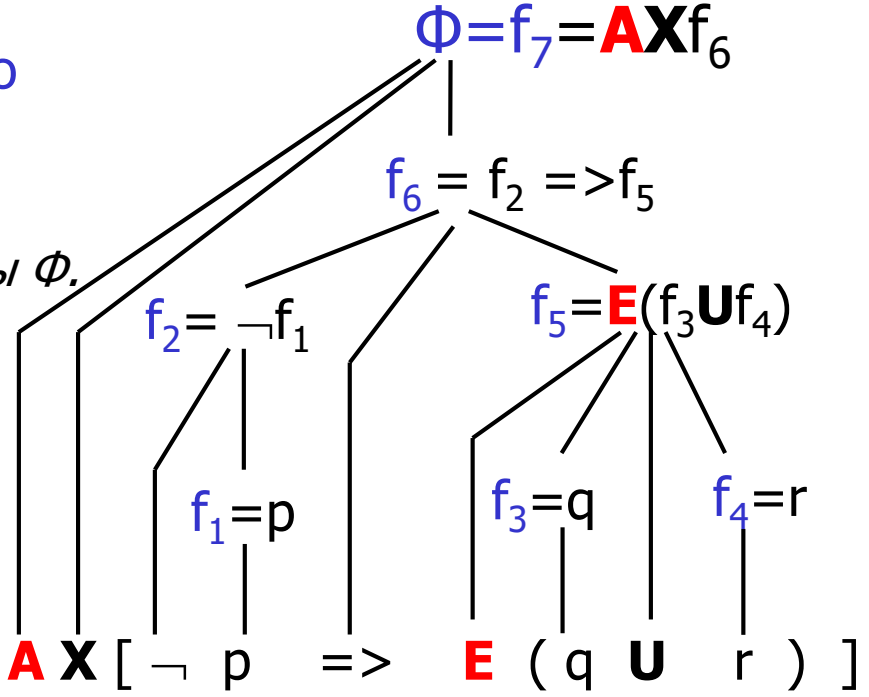
Общая идея – структурная индукция:

Формул – бесконечное число, но число типов подформул конечно! Они перечислены в ГРАММАТИКЕ языка

а) *Строим синтаксическое дерево формулы Ф.*

б) *Последовательно помечаем (маркируем) все состояния структуры Крипке теми подформулами формулы Ф, которые истинны в этих состояниях.*

в) *По окончании алгоритма, если начальное состояние структуры Крипке М помечено Ф, то Ф выполняется на М.*



Необходимо разработать алгоритмы маркировки состояний структуры Крипке для каждой возможной подформулы формулы Ф

Подформулы CTL формул

Все возможные подформулы задаются грамматикой

Синтаксис (грамматика):

$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{AX}\varphi \mid \mathbf{EX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{AG}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{A}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{E}[\varphi_1 \mathbf{U} \varphi_2]$

Алгоритм для каждого типа формулы φ должен определять, выполняется ли формула φ в каждом состоянии структуры Крипке в том предположении, что выполнение или невыполнение подформул φ уже определено во всех состояниях.

Обозначим: $s \models \varphi$ утверждение:

"в состоянии s структуры Крипке формула φ выполняется".

$s \models p$ если и только если состояние s *ПОМЕЧЕНО* атомарным предикатом p.

$s \models \neg\varphi$ если и только если в состоянии s НЕ выполняется формула φ.

$s \models \varphi_1 \vee \varphi_2$ если и только если в s выполняется или φ_1, или φ_2.

$s \models \mathbf{EX}\varphi$ если и только если из s *существует путь*, в следующем состоянии которого выполняется φ.

Это, фактически, неформальное определение семантики формул CTL.

Можно строить алгоритмы не для всех типов подформул CTL. Базисы CTL

Синтаксис (грамматика):

$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{AX}\varphi \mid \mathbf{EX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{AG}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{A}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{E}[\varphi_1 \mathbf{U} \varphi_2]$

Нужны ли все эти конструкции?

1. Очевидно $AX\varphi \equiv \neg E \neg X\varphi$. Используем, $\neg X\varphi \equiv X\neg\varphi$. Поэтому $AX\varphi \equiv \neg EX\neg\varphi$
   Отсюда: AX и EX – дуальны, взаимозаменяемы.

2. Имеем соотношения: $A\Phi \equiv \neg E \neg \Phi$; $E\Phi \equiv \neg A \neg \Phi$
   Поэтому $AF\varphi \equiv \neg E \neg F\varphi$. Но полученная формула не является формулой CTL, нужны только комбинации *<квантор пути, темпоральный оператор>*.

Используем следующее соотношение:

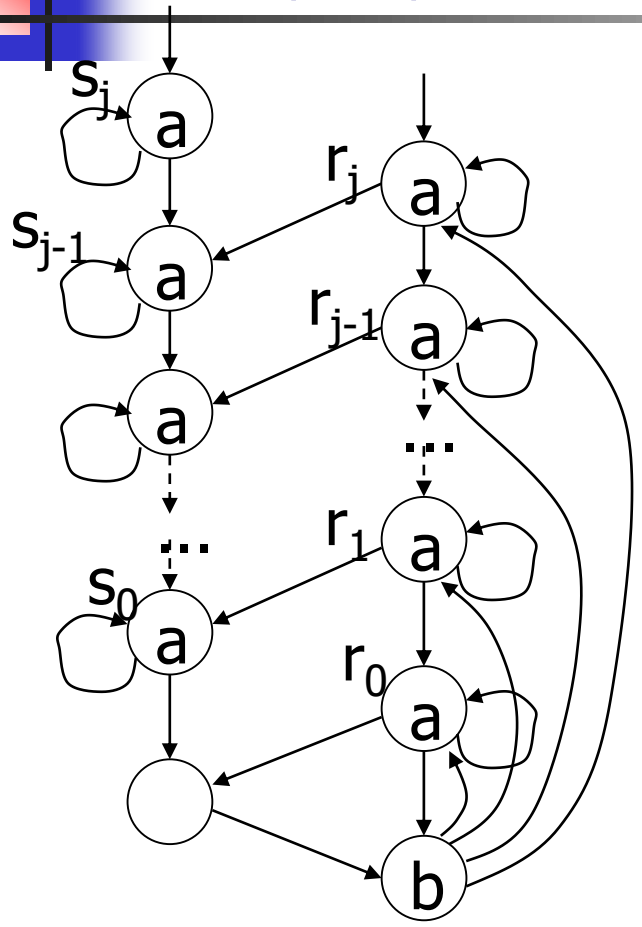
$F\varphi \equiv \neg G \neg \varphi$. Отсюда, $AG\varphi \equiv \neg E \neg G\varphi \equiv \neg EF\neg\varphi$, т.е. AG можно заменить на EF, и наоборот. Поэтому AG и EF дуальны, взаимозаменяемы. AG φ заменяем на $\neg EF\neg\varphi$

3. Аналогично, EG и AF – дуальны, взаимозаменяемы.

4. Известно: $F\varphi \equiv True U\varphi$, поэтому EF можно заменить на E*True*U, а AF – на A*True*U.

5. НО! Оказывается: $A(\varphi_1 U \varphi_2) \equiv AF\varphi_2 \wedge \neg E(\neg\varphi_2 \, U(\neg\varphi_1 \wedge \neg\varphi_2))$.
   Т.е. AU можно выразить через AF и EU.

О EU, AU, EF



Структура Крипке для доказательства теоремы

Теорема. EU не может быть выражено с помощью EF и AU

Рассмотрим подъязык L_A

$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{EX}\ \varphi \mid \mathbf{A}(\varphi_1 \mathbf{U} \varphi_2) \mid \mathbf{EF}\ \varphi$

Доказывается, что не существует формулы φ из языка L_A глубиной $k \le j$ для любого $j>=0$, которая различала бы состояния s_j и r_j для структуры Крипке на примере.

Рассматривается индукция по глубине формулы.
Для $k=0$ очевидно для любых s_i, r_i.
Для $k=k'-1$ предполагаем, что такой формулы нет
Для k доказывается структурной индукцией по грамматике L_A

Однако формула CTL φ=E(aUb) различает состояния s_j и r_j

Взаимозависимости комбинаторов CTL

Взаимозависимости CTL формул

$\mathbf{AX}\varphi \equiv \neg\mathbf{EX}\neg\varphi$

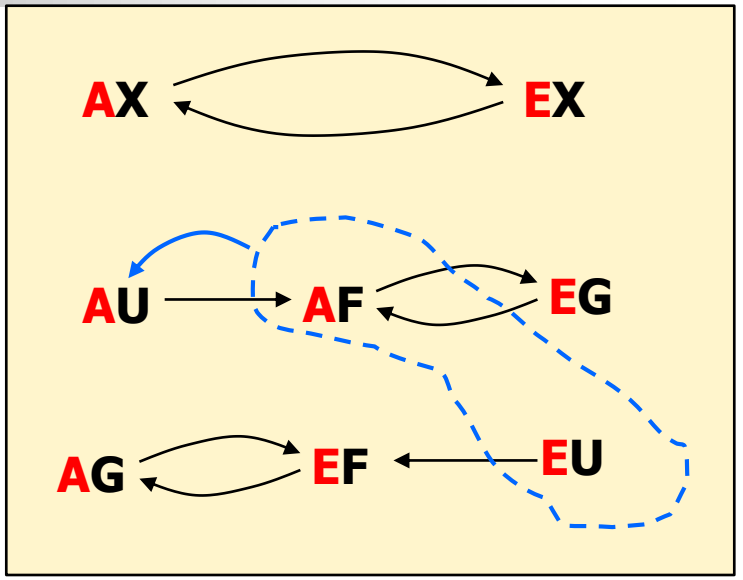
$\mathbf{EG}\varphi \equiv \neg\mathbf{AF}\neg\varphi$

$\mathbf{AG}\varphi \equiv \neg\mathbf{EF}\neg\varphi$

$\mathbf{AF}\varphi \equiv \mathbf{A}(\ \mathit{True}\ \mathbf{U}\ \varphi\ )$

$\mathbf{EF}\varphi \equiv \mathbf{E}(\ \mathit{True}\ \mathbf{U}\ \varphi\ )$

$\mathbf{A}(\varphi_1\mathbf{U}\varphi_2\ ) \equiv \mathbf{AF}\varphi_2 \wedge \neg\mathbf{E}(\neg\varphi_2\ \mathbf{U}(\neg\varphi_1\wedge\neg\ \varphi_2\ ))$



Возможные базисы CTL

Всего существует 6 базисов CTL

{ **EX**, **AF**, **EU** },
{ **EX**, **A*True*U**, **EU** },
{ **EX**, **EG**, **EU** } и т.д.

Для процедуры Model checking достаточно построить алгоритмы маркировки только для CTL-комбинаторов какого-нибудь базиса.
Такие алгоритмы просты для EX, AX, EF, AF, EU, A*true*U.
Для EG и AG алгоритмы чуть сложнее, но эффективнее.

Алгоритм Model Checking: базис {EX, AF, EU}

```
for all 0<i≤|Φ| do          /* для всех уровней синтаксического дерева */
  for all Ψ ∈ Sub(Φ) with |Ψ| = i do /* для всех подформул уровня i*/
    switch(Ψ):              /* для каждого типа подформул – свой алгоритм */
      p        :   Sat_Ψ := { s∈S| p∈L(s) };
      ¬p       :   Sat_Ψ := { s∈S| p∉L(s) };
      p∧q      :   Sat_Ψ := { s∈S| p,q ∈L(s) };
      EXp      :   Sat_Ψ := Sat_EX(p);
      AFp      :   Sat_Ψ := Sat_AF(p);
      E(pUq)   :   Sat_Ψ := Sat_EU(p,q);
    end switch   /* Sat_EX,Sat_AF и т.п.-функции,
                                определенные далее */
    for all s∈Sat_Ψ do L(s):= L(s)∪{p_Ψ} od
      /* Все состояния из Sat_Ψ помечаем новым атомарным предикатом p_Ψ*/
  od;
od
```

Sat_Ψ - множество состояний, в которых выполняется формула Ψ

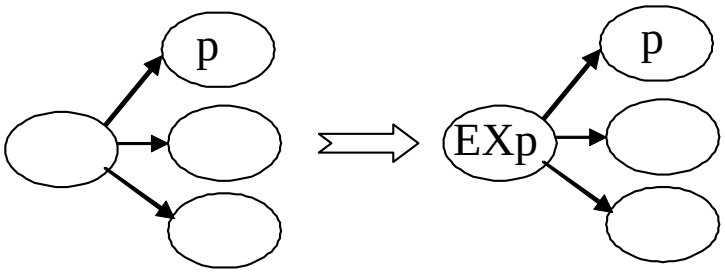
L(s) - множество (атомарных) формул, которые выполняется в состоянии s

Алгоритм обрабатывает его все узлы синтаксического дерева формулы Ф.

Если φ выполняется в НАЧАЛЬНЫХ состояниях M, то она выполняется на M

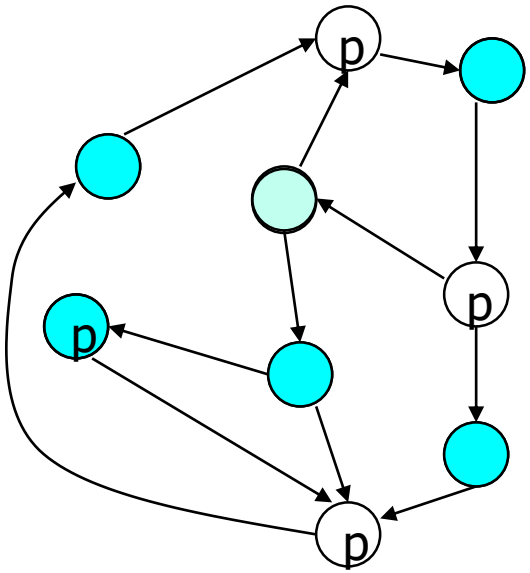
Алгоритм Model Checking для CTL (EX)

EXp: помечаем s меткой p_{EXp} если хотя бы один преемник s помечен p



было стало

```
function SAT_EX(p)  /*дает все s,
                      в которых истинна EXp */
    local var Y;
    begin
          Y:= { s| (∃s1 ∈ SAT (p)) s→s1 };
        return Y
    end
```



Все закрашенные состояния попадают в множество SAT_EX(p)

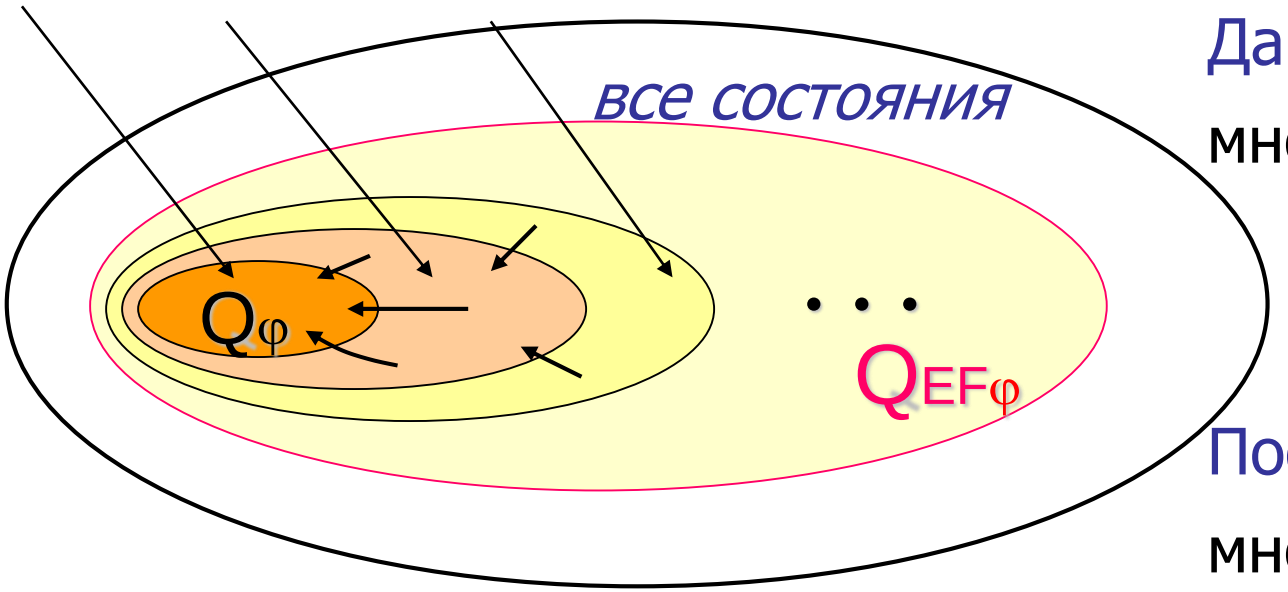
Перебираем все состояния структуры Крипке и смотрим, есть ли из текущего состояния ребро в состояние, помеченное p

Вычисление множеств состояний, помеченных операторами базиса: EFφ

$EF\varphi = \varphi \vee EX\varphi \vee EX\ EX\varphi \vee EX\ EX\ EX\varphi \vee \ldots$

$Q_{EF\varphi}$ = *состояния из которых существует путь, по которому можно достичь состояния, помеченного* φ *- это состояния, помеченные*:

φ $EX\varphi$ $EX(\ EX\varphi\ )$ ...



Дано:

множество Q_{φ}.

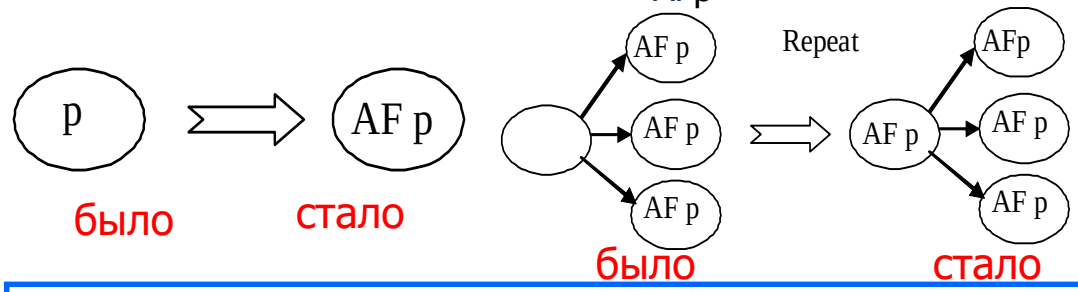
Построить:

множество $Q_{EF\varphi}$.

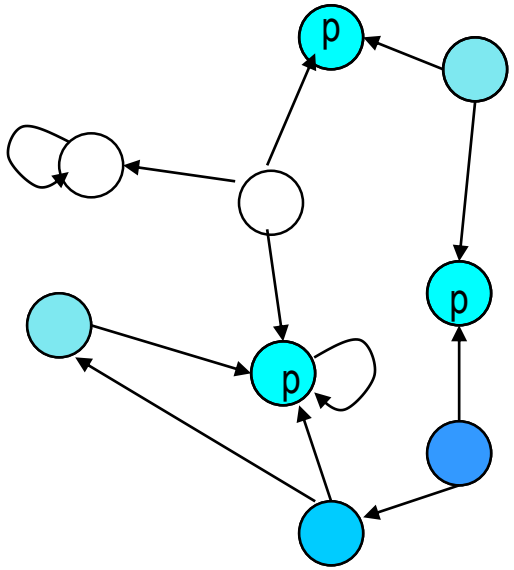
Построение множества $Q_{EF\varphi}$ останавливается, когда объединение множеств перестает расти

Алгоритм Model Checking для CTL (AF)

AFφ: каждое состояние, если оно помечено p помечаем p_{AFp};
повторяем: s помечаем p_{AFp} если все преемники s помечены p_{AFp}




```
function SAT_AF(p)
        /*дает все s, в которых истинна AFp */
    local var X,Y
    begin
        X:=S;
        Y:= SAT (p);
        repeat until X=Y
        begin
           X:=Y;
           Y:= Y ∪ { s | (∀s1:s→s1) s1∈Y }
         end
         return Y
     end
```

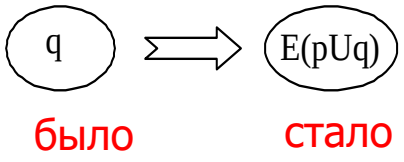


Все закрашенные состояния попадают в множество

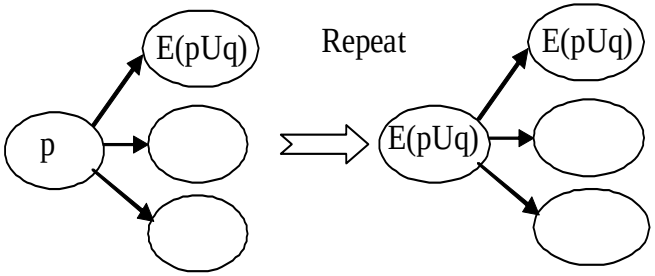
SAT_{AFp}

Алгоритм Model Checking для CTL (EU)

E(pUq): помечаем состояние меткой $p_{E(pUq)}$, если оно уже помечено q; повторяем: помечаем s меткой E(pUq) если оно помечено p и хотя бы один преемник s помечен $p_{E[pUq]}$

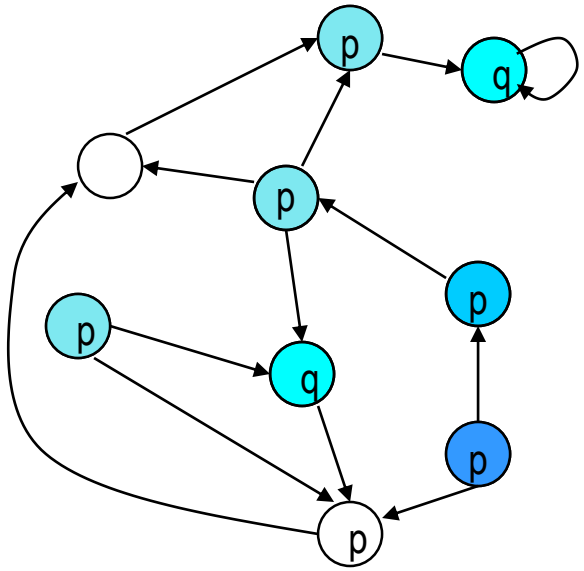


было стало



было стало

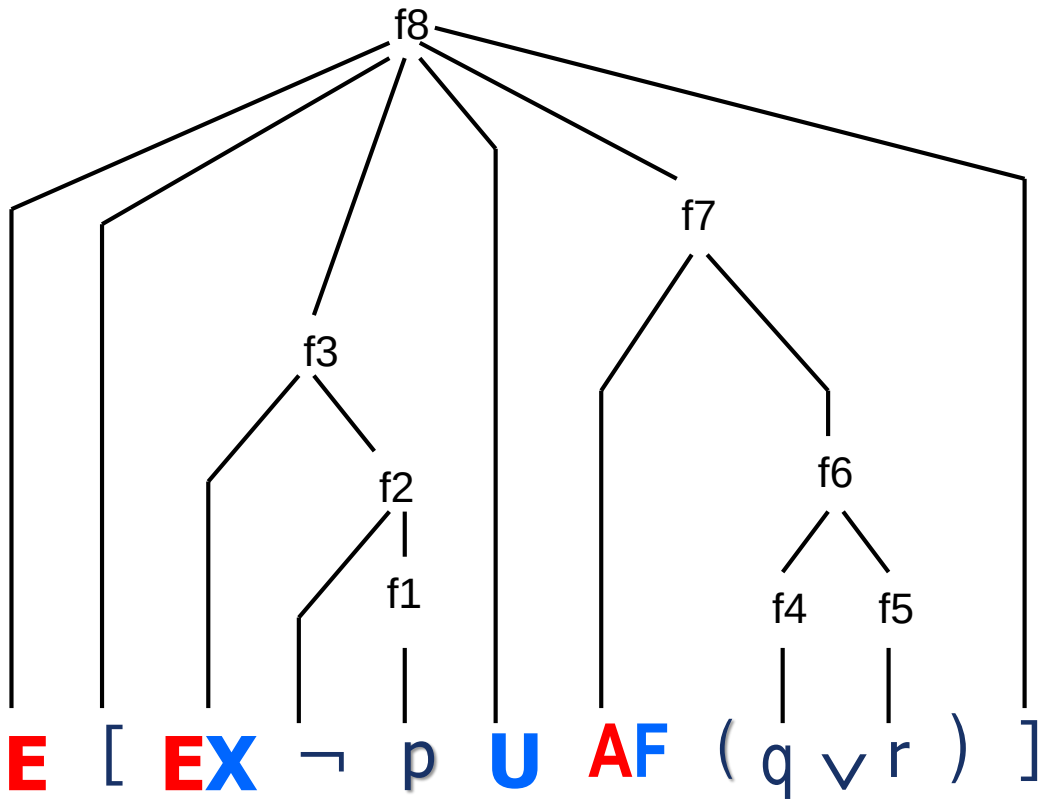
```
function SAT_EU (p, q) /*дает все s, удовл E(pUq) */
    local var P, X, Y
    begin
        P:= SAT (p);  X:=S;  Y:= SAT (q);
        repeat until X=Y
        begin
           X:=Y;
           Y:= Y ∪ (P ∩ { s | (∃s1∈Y) s→s1 } )
         end
         return Y
    end
```



Все закрашенные состояния попадают в множество $SAT_{E(pUq)}$

Model Checking: синтаксический анализ формулы

Пример: ϕ = E [EX$\neg$p U AF (q $\vee$ r)]



Последовательно снизу вверх:

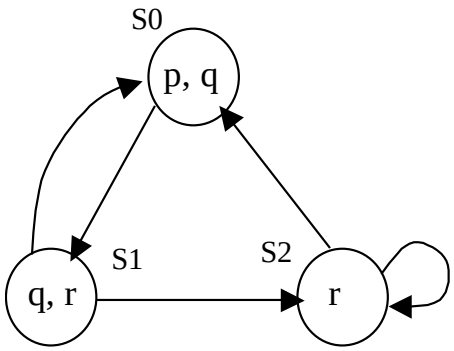
- f1 = p;
- f2 = $\neg$f1;
- f3 = EX f2;
- f4 = q ;
- f5 = r ;
- f6 = f4 $\vee$ f5;
- f7 = AF f6;
- f8 = E[f3 U f7].

Обычный синтаксический анализ

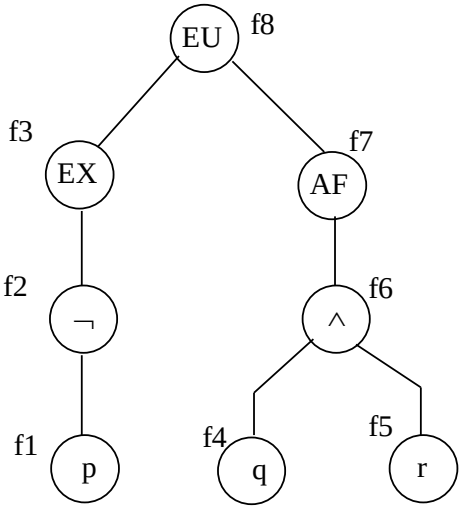
Model Checking – пример

f8 выполняется в s_0, s_1, s_2

$\Phi = E\,[EX\neg p\ U\ AF\,(q \wedge r)\,]$



Синтаксическое дерево:



f1 = p

f2 = ¬f1

f3 = EX f2

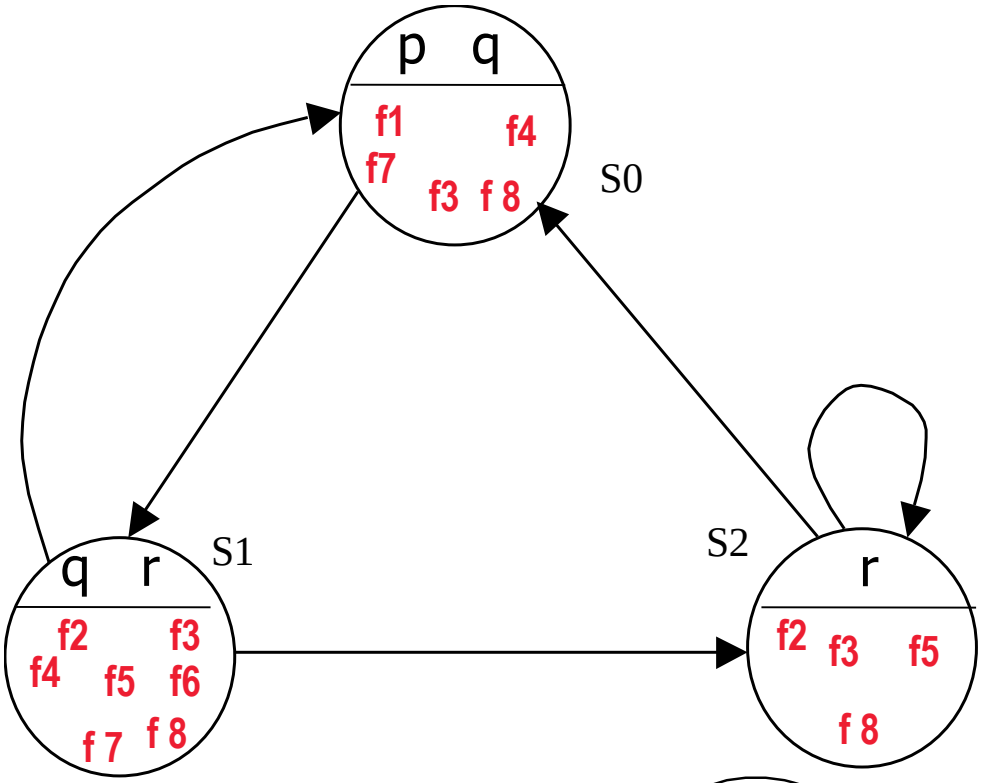
f4 = q

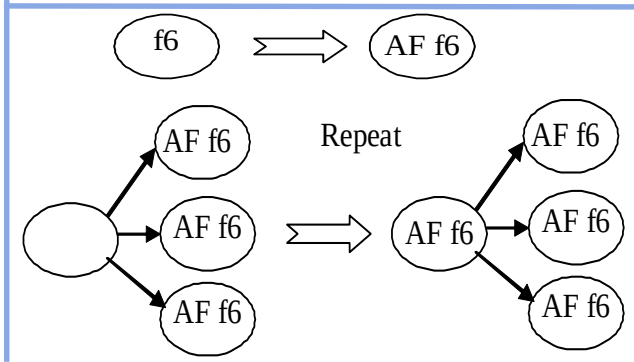
f5 = r

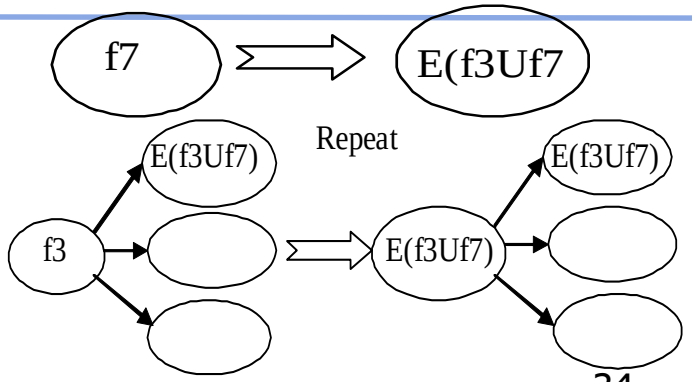
f6 = f4 ∧ f5

f7 = AF f6

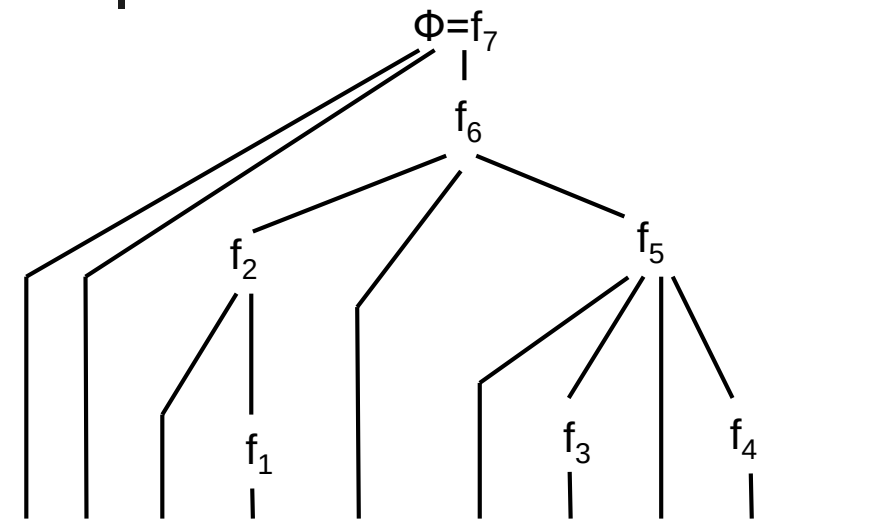
f8 = E[f3 U f7]







Алгоритм маркировки для CTL формул



$f_1 = p$

$f_2 = \neg f_1$

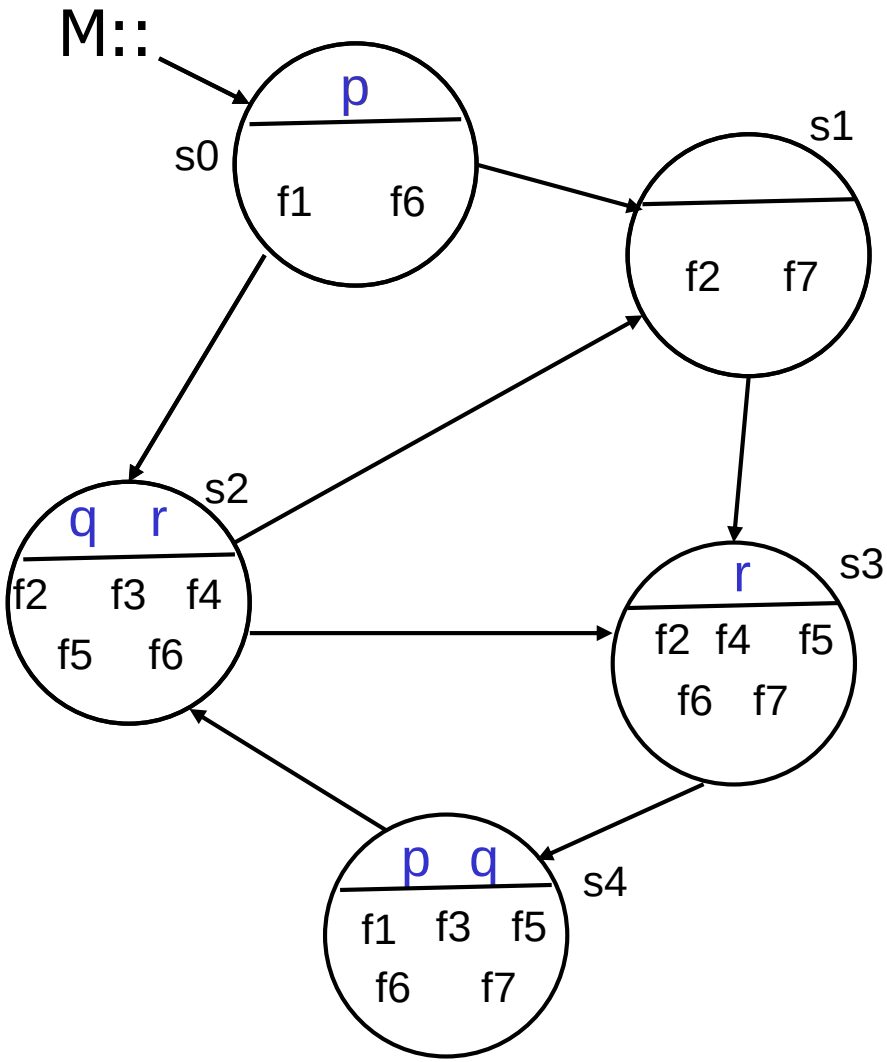
$f_3 = q$

$f_4 = r$

$f5 = E(f3\ U\ f4)$

$f6 = f2 \Rightarrow f5$

$f7 = \Phi = AX\ f6$

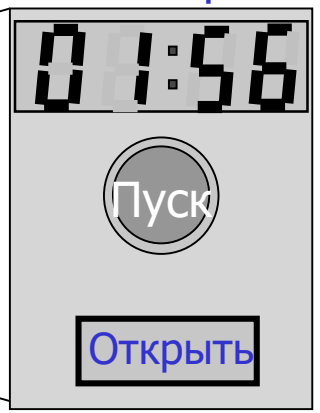


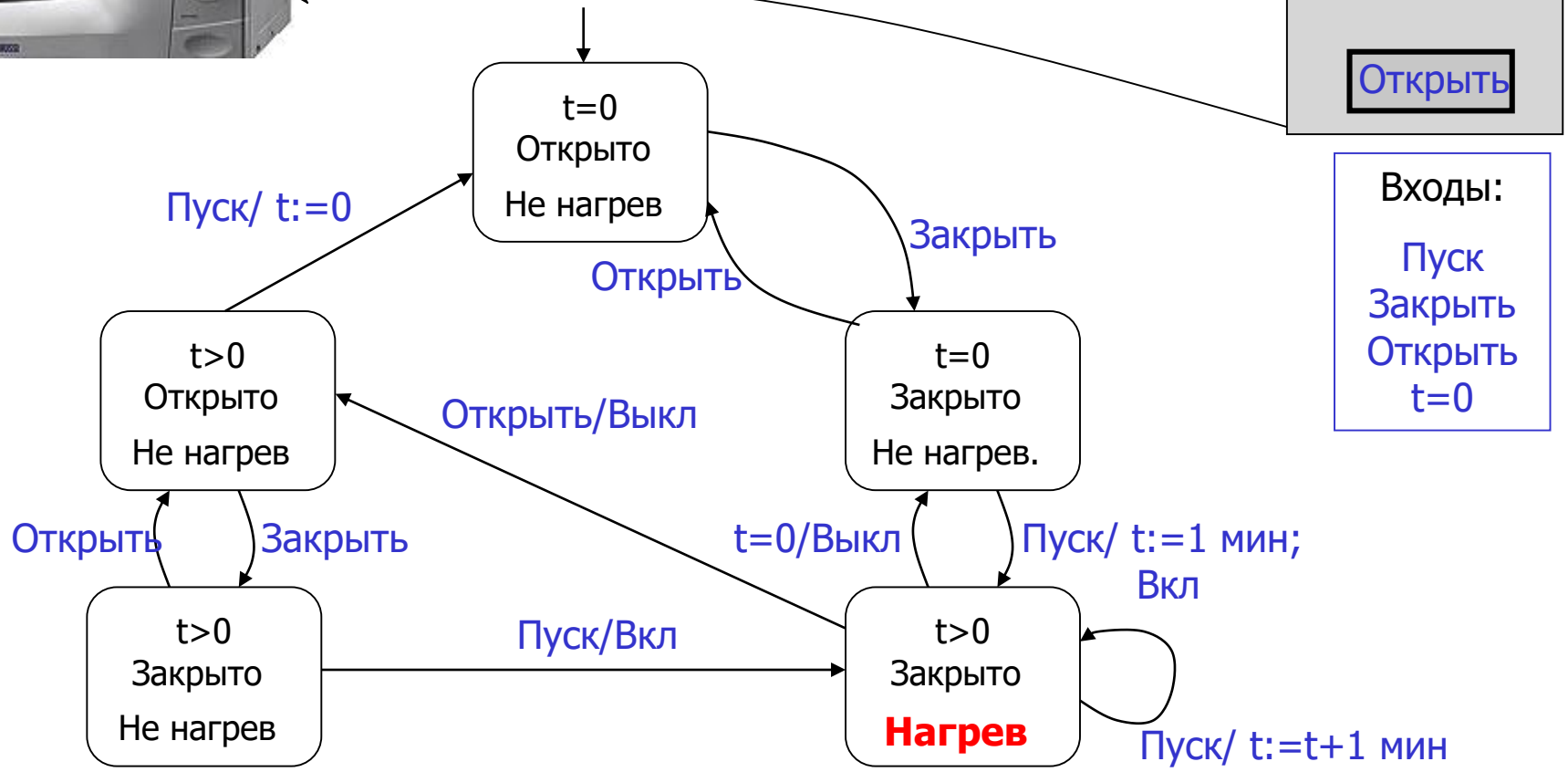
На M **не** выполняется Ф

Система управления микроволновой печью

Redmond RM-1004

Таймер



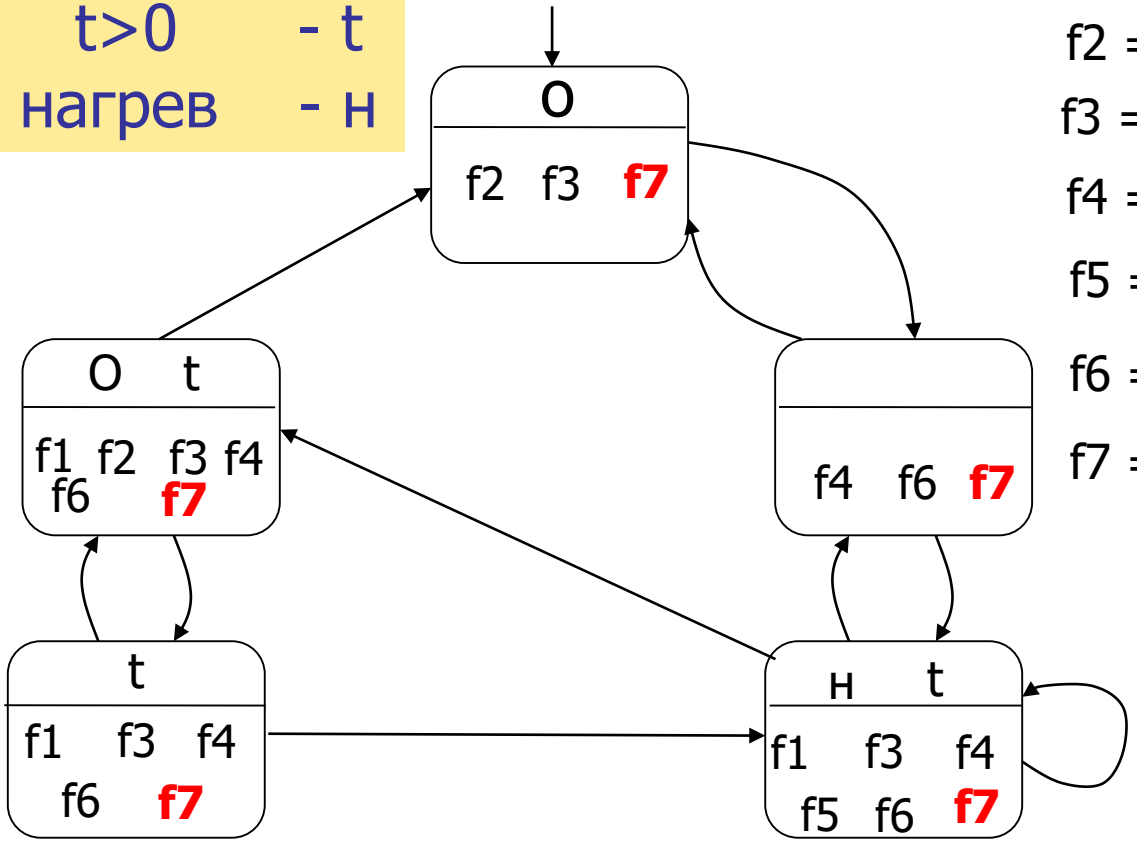


Входы:

Пуск
Закрыть
Открыть
t=0

Верификация системы управления микроволновой печью

открыто – о
t>0 - t
нагрев - н



f1 = t

f2 = о

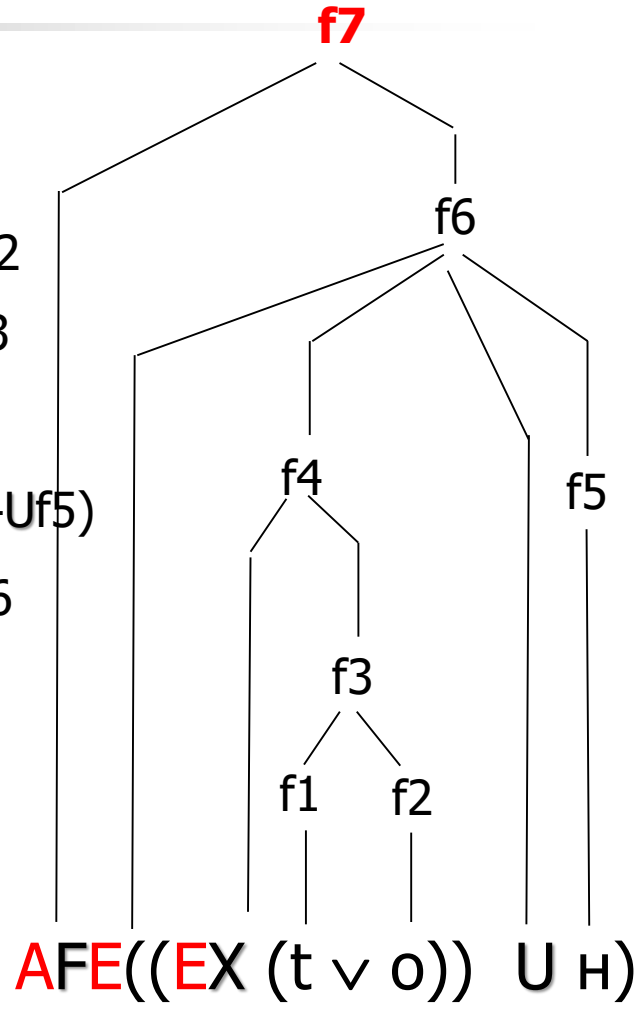
f3 = f1∨f2

f4 = EXf3

f5 = н

f6 = E(f4Uf5)

f7 = AFf6



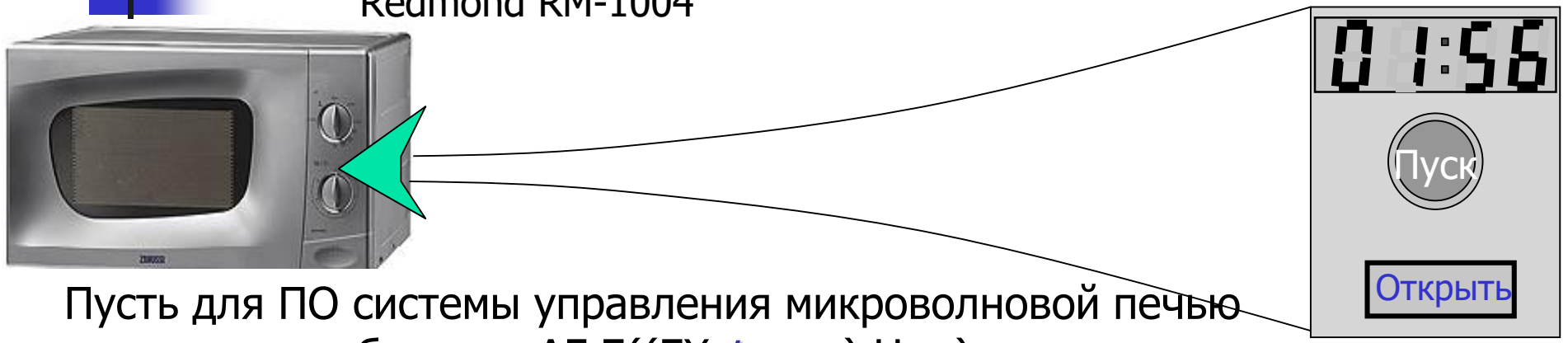
Задача: проверить выполнение свойства: **AF E** ((**EX** (t ∨ о)) **U** н)

Ответ: формула **AF E** ((**EX** (t ∨ о)) **U** н) выполняется

Основная идея верификации методом Model checking

открыто – o
t>0 - t
нагрев - н

Redmond RM-1004



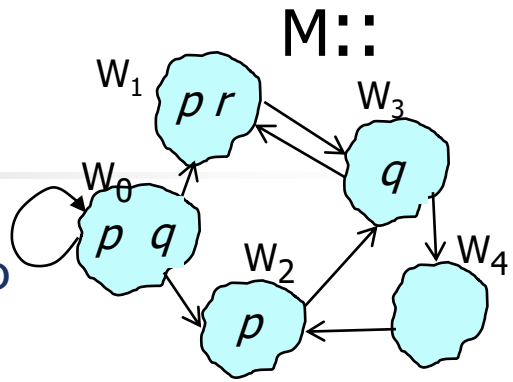
Пусть для ПО системы управления микроволновой печью проверено: требование AF E((EX *t* ∨ *o*) U *н*) выполняется.

Можно ли утверждать, что эта система управления корректна вообще (правильна)? **НЕТ!**

Мы ДОКАЗАЛИ только, что при функционировании системы ошибки конкретного типа не возникнут.

Относительно ошибок другого типа мы сказать ничего не можем! Нужно проверять!

Резюме: Алгоритм Model Checking для CTL (алгоритм пометок)



Вход: Структура Крипке M=(S, →, L) и CTL формула Φ

Выход: Множество состояний, удовлетворяющих формуле Φ

1 шаг: Для формулы Φ выполняем синтаксический анализ

2 шаг: Все состояния M помечаем новыми атомарными предикатами, для подформул Φ, начиная с внутренних, которые в этих состояниях истинны

Если эта подформула:

p : помечаем каждое состояние s атомом p, если p∈L(s)

¬p : помечаем каждое s атомарным предикатом ¬p, если s не помечена p

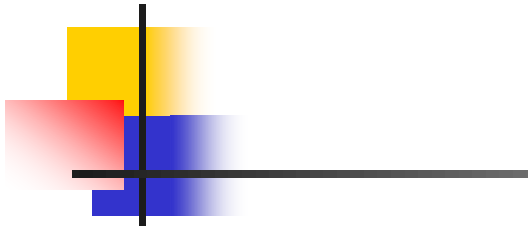
p ∨ q: помечаем каждое s атомарным предикатом p∨q если s помечена p или q

AFp : помечаем каждое s новым атомом AFp, если оно помечено p;
в цикле: помечаем каждое s атомом AFp, если все преемники s помечены AFp

E[pUq] : помечаем s новым атомом E[pUq], если оно помечено q; в цикле: помечаем каждое s атомом E[pUq], если s помечено p и хотя бы один преемник s помечен E[pUq]

EXp : помечаем каждое s атомом EXp, если хотя бы один преемник s помечен p

Сложность алгоритма O[|Φ|*| S|*(|S| + | T|)].

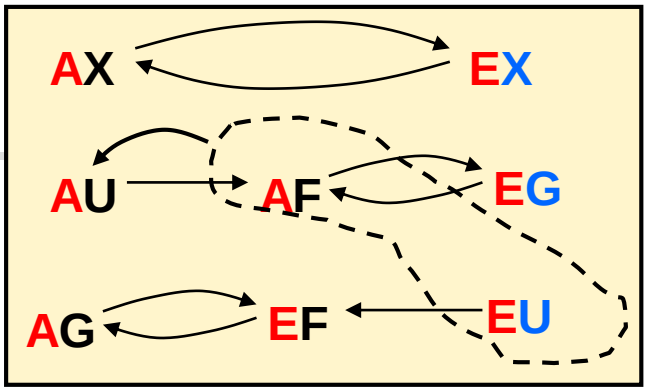


- Для заданных структуре Крипке и формулы Ф логики CTL нужно уметь строить множества состояний, в которых выполняется формула Ф (и все ее подформулы).
- Это является стандартным умением, которое будет проверяться у каждого студента.

Базисы CTL

Синтаксис (грамматика):

$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{AX}\varphi \mid \mathbf{EX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{EF}\varphi \mid$
$\mathbf{AG}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{A}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{E}[\varphi_1 \mathbf{U} \varphi_2]$



- Базисы CTL:
  - EX, EG, EU
    - $AX\varphi \equiv \neg EX\neg\varphi$,
    - $EF\varphi = E[TU\varphi]$,
    - $AG\varphi \equiv \neg EF\neg\varphi$,
    - $AF\varphi \equiv \neg EG\neg\varphi$,
    - $A[\varphi U\psi] \equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge AF\psi$

    Выражение других пар через базис
  - EX, A*True*U, EU
    - $AF\varphi \equiv A[T U\varphi]$,
    - $EF\varphi \equiv E[T U\varphi]$,
    - ...

Удобно рассматривать как базис только из кванторов существования

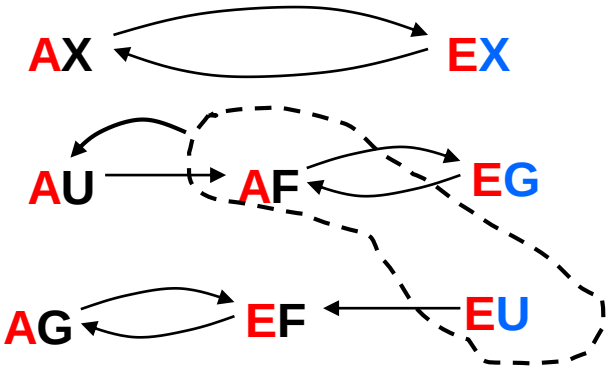
Синтаксис (грамматика):

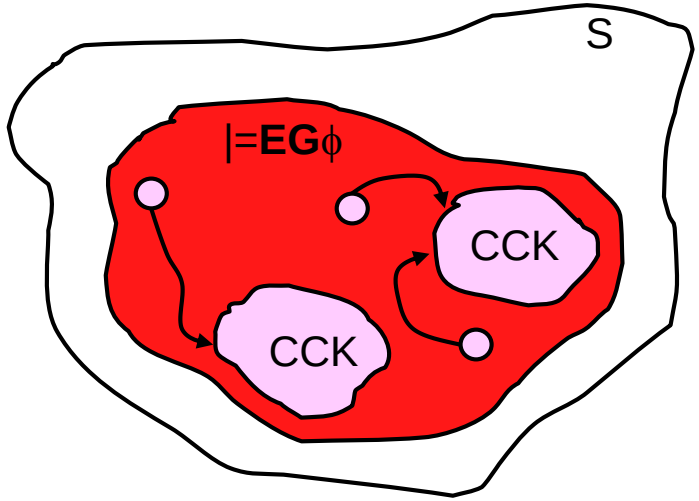
$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid$
$\mathbf{EX}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{E}[\varphi \mathbf{U} \varphi]$

Этот базис требует только простых алгоритмов

Model Checking алгоритм (более эффективный)

Трансляция Ф в базис {¬, ∨, EG, EU, EX}





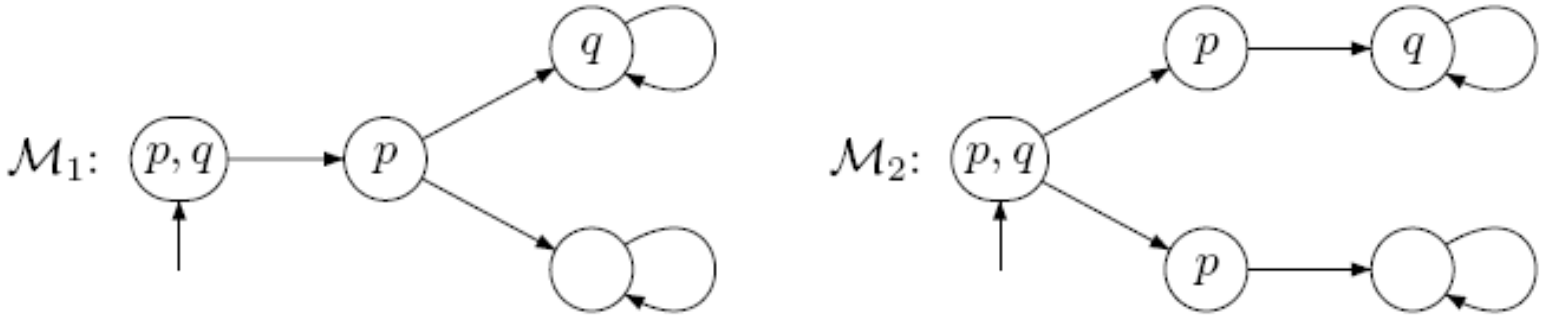
EGϕ:

1. Выбрасываем из M все состояния, которые не помечены ϕ
2. Находим все сильно связные компоненты (ССК) в оставшемся графе
3. Обратный поиск всех состояний, которые связаны с любой ССК

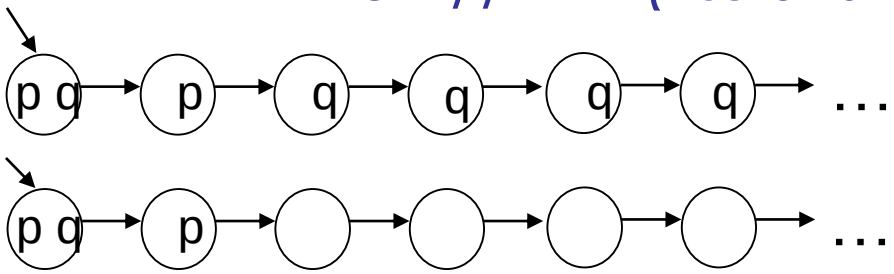
Сложность алгоритма O[|Ф| *(| K |)]

- [Clarke & Emerson, 1981, Queille& Sifakis, 1982]: Model checking programs of size m wrt CTL formulas of size n can be done in time mn.

Различия темпоральных логик LTL и CTL



У этих двух структур Крипке одно и то же множество возможных ПОВЕДЕНИЙ (всего таких поведений ДВА!):

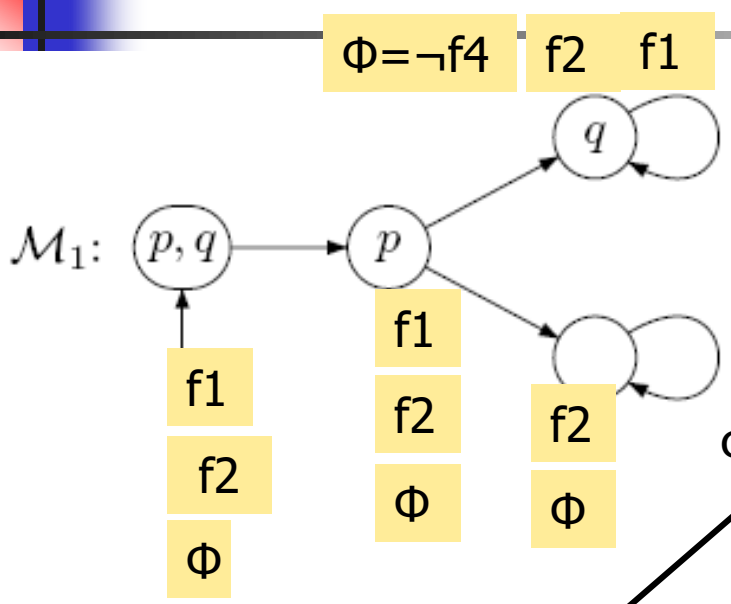


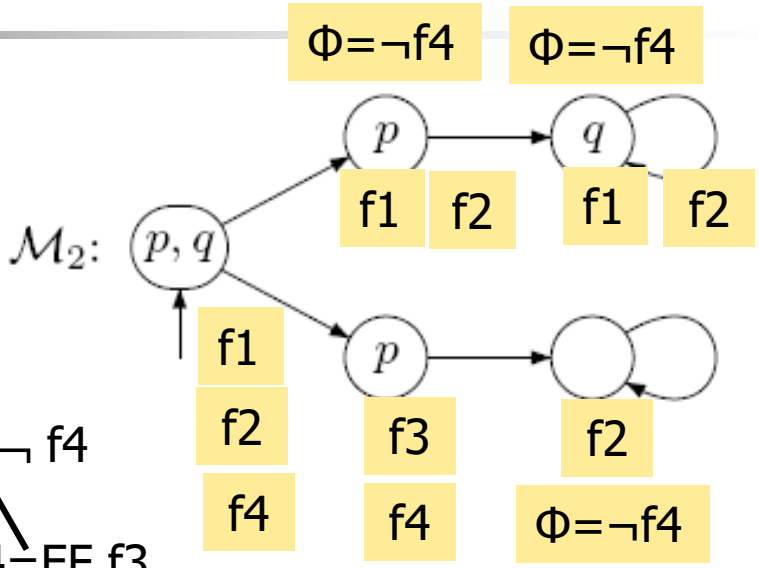
Но их деревья поведений различны!

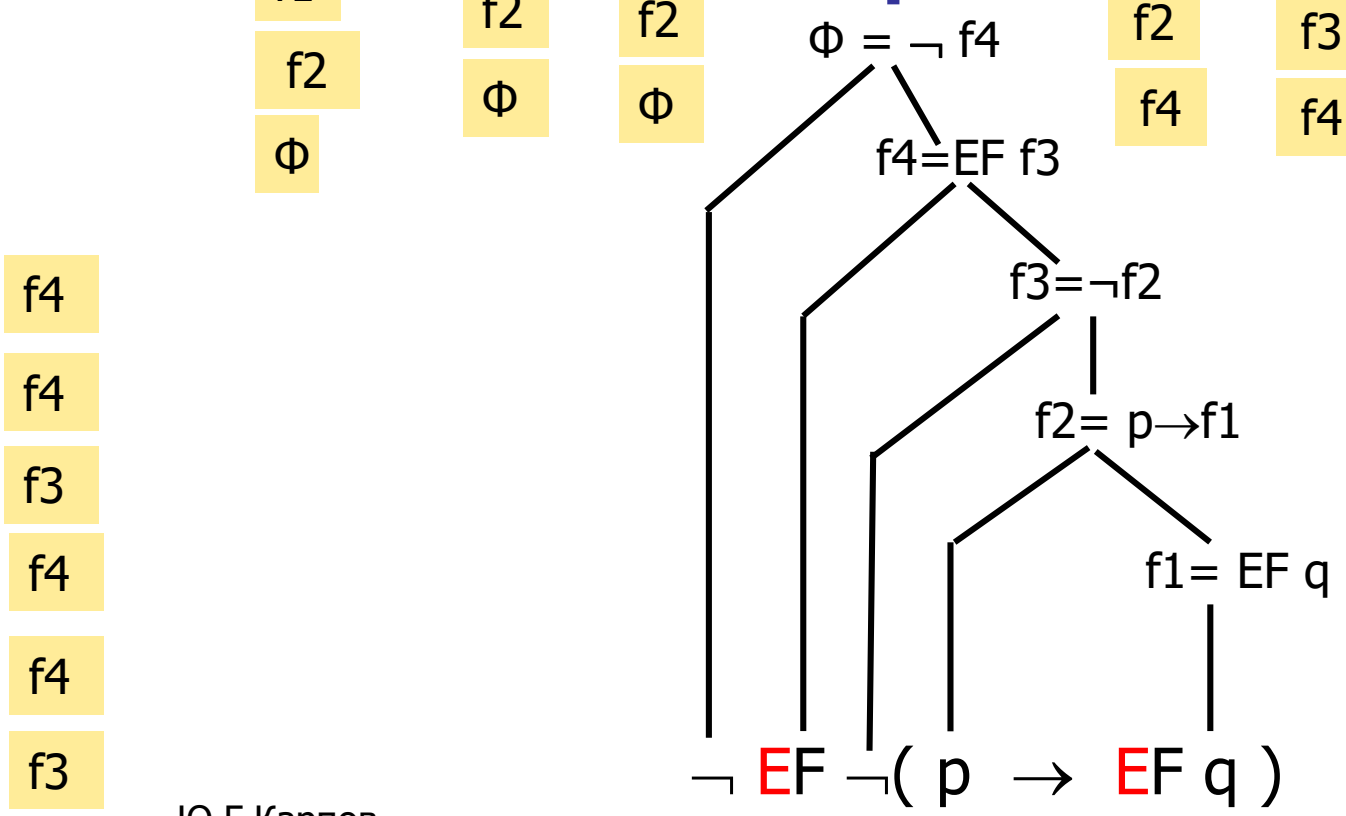
- Модели M_1 и M_2 нельзя различить никакой формулой LTL (*потому что формулы LTL описывают **линейные** ПОВЕДЕНИЯ*).
- M_1 и M_2 различаются формулой логики CTL $AG(p \rightarrow EFq)$.

- AG(p→EFq)

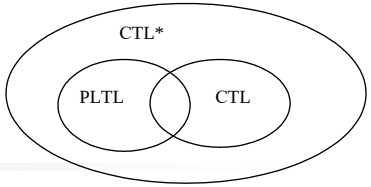
$AG\varphi \equiv \neg E \neg G\varphi \equiv \neg EF \neg \varphi$







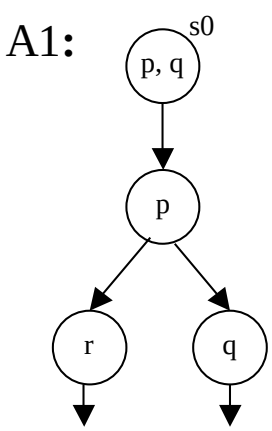
Сравнение логик LTL, CTL и CTL* (семантика)

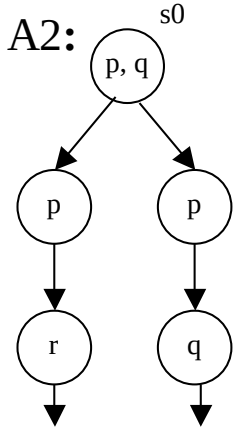


E[pU(qUr)] - не CTL-формула, но: E[pU(qUr)] = E[pUE(qUr)]

Выполняется ли соотношение между темпоральными логиками и **семантически**?

Теорема. Существуют свойства поведений, выражаемые в CTL и не выражаемые в LTL





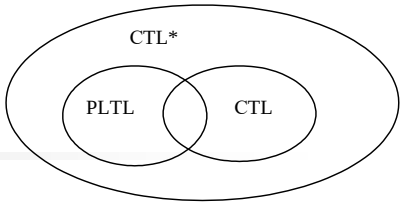
Доказательство. поведения A1 и A2 неразличимы в LTL, каждая имеет две одинаковые траектории:
{p,q}, {p}, {r},... и {p,q}, {p}, {q},...

В CTL можно выразить, что выбор между r и q в A1 сохраняется дольше:
A1, s0 |= AX(EXq & EXr),
A2, s0 |≠ AX(EXq & EXr)

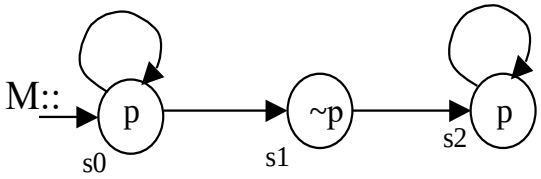
Теорема. Существуют свойства поведений, выражаемые в LTL и не выражаемые в CTL.

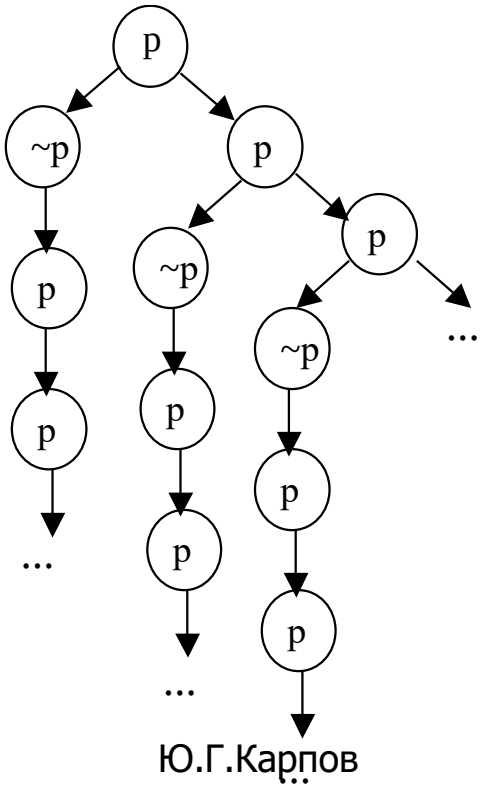
Доказательство. Свойство AFGp логики LTL не выражается в логике CTL

Пример: свойство LTL, не выражаемое в CTL



Теорема. Существуют свойства поведений, выражаемые в LTL и не выражаемые в CTL.





Пример. В CTL нельзя выразить свойство FG

Каждый прогон (run) системы M удовлетворяет FGp, т.е. когда-нибудь в будущем на каждой траектории будет Gp. Т.е. M |= AFGp

НО! M |≠ AF AGp, поскольку та траектория, которая всегда остается в s0, может в любой момент перейти в состояние s1, в котором ¬p

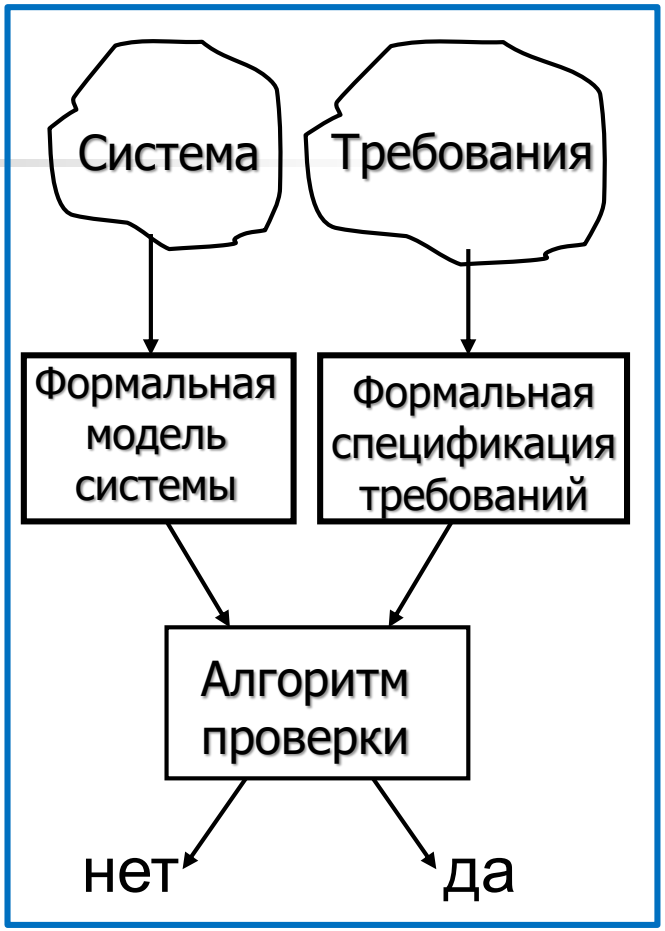
Следовательно, AFGp ≠ AFAGp

CTL позволяет выразить свойство достижимости по пути, но не позволяет выразить другие свойства, которые могут встретиться вдоль пути.

Доказательство корректности программ. Современное состояние

Система корректна, если она соответствует требованиям, установленным при ее разработке

Технология верификации Model checking
позволяет для любой программной (аппаратной) системы
с конечным числом состояний
проверить выполнение ЛЮБОГО требования к ее ПОВЕДЕНИЮ, выраженного формулой темпоральной логики (LTL или CTL)
– но только если хватит ресурсов



Мы можем говорить о корректности программы *ТОЛЬКО* относительно конкретных требований к ее поведению.

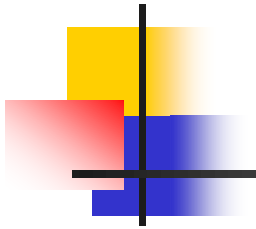
Какие это требования? – *КАКИЕ УГОДНО, выраженные формулами TL!*

Сколько требований мы можем проверить? – *СКОЛЬКО УГОДНО!*

Как выбирать проверяемые требования? – *Мы этим займемся.*

Заключение

- Программная система функционирует правильно (т.е. она корректна) не тогда, когда не удалось найти в ней ошибок, а тогда, когда удалось ДОКАЗАТЬ ОТСУТСТВИЕ В НЕЙ ОШИБОК.
- С помощью техники Model checking мы можем ДОКАЗАТЬ, что при функционировании системы не возникнут ошибки конкретного типа (те ошибки, которые нарушают свойство, выраженное конкретной формулой TL).
- Относительно того, возникнут ли ошибки ДРУГОГО типа, мы НИЧЕГО сказать не можем. Нужно выполнить проверку другой формулы темпоральной логики.
- **ИСЧЕРПЫВАЮЩЕГО ДОКАЗАТЕЛЬСТВА отсутствия ВСЕХ ВОЗМОЖНЫХ** ошибок в программной (или аппаратной) системе этот метод не дает.
- Проверка того, что данная формула Ф темпоральной логики CTL ИСТИННА на данной структуре Крипке, осуществляется последовательным вычислением истинности всех подформул формулы Ф на всех состояниях структуры Крипке.
- Сложность алгоритма проверки того, является ли структура Крипке К моделью CTL-формулы Ф, удивительно мала: O(|K| * |Ф|).
- Model checking является ПРОРЫВОМ в верификации программ.



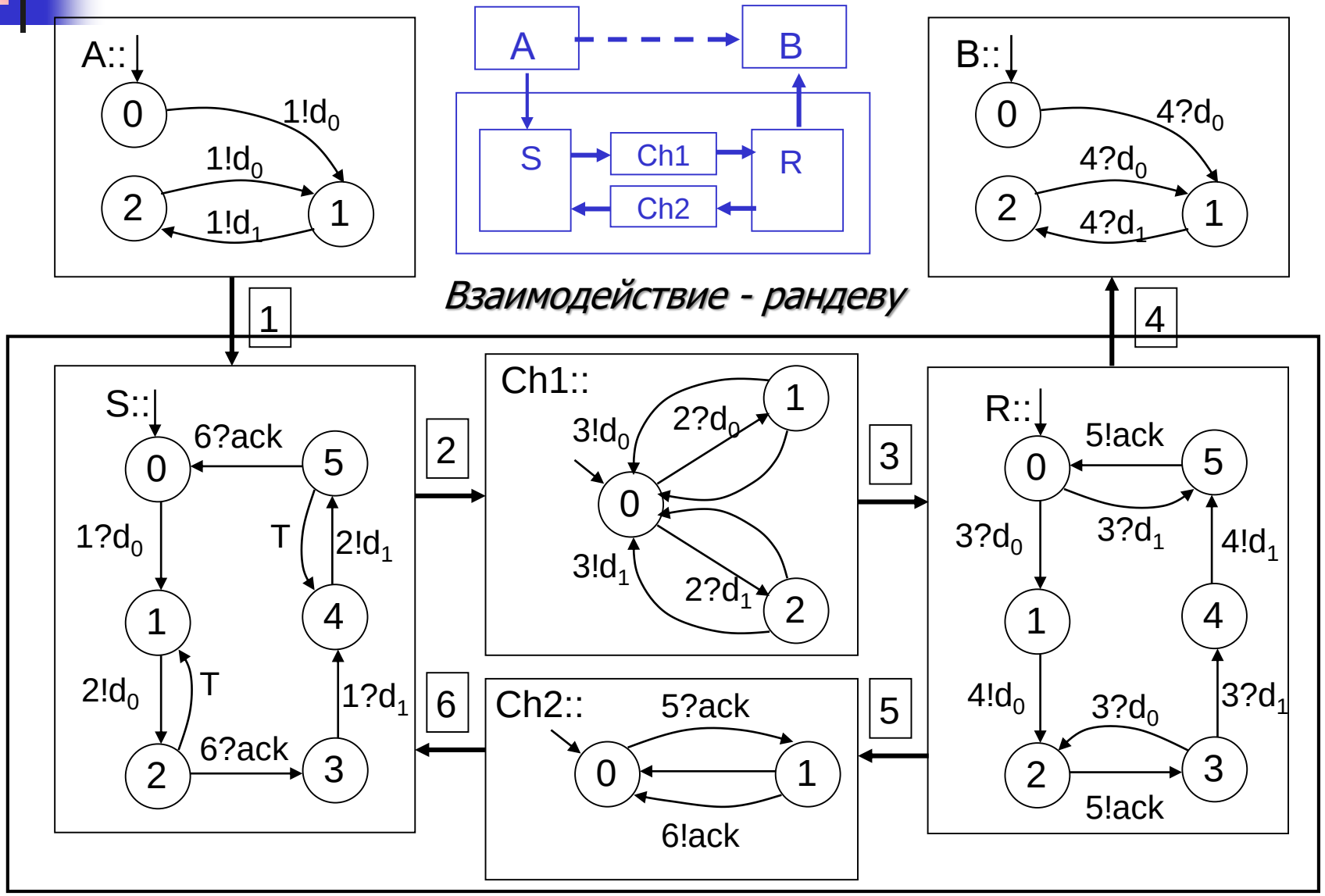
Спасибо за внимание

Параллельные системы, специфицированные в виде взаимодействующих КА

Архитектура протокола PAR

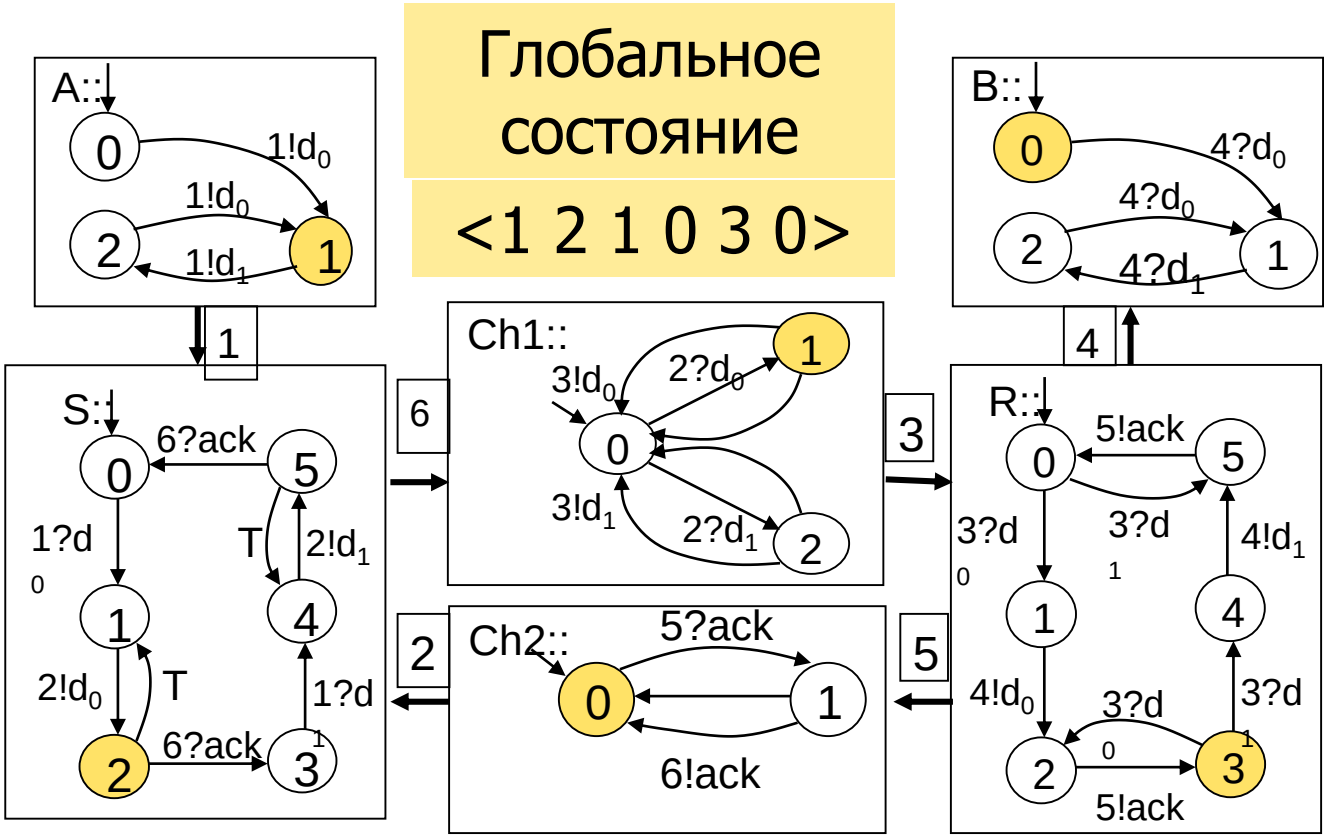
Процесс А использует протокол коммуникации для передачи потока сообщений чрез НЕНАДЕЖНУЮ СРЕДУ процессу В. Он отдает передатчику S по его готовности очередное сообщение, а процесс В получает поток посланных сообщений от приемника R. Если сообщения в сети теряются или задерживаются, то сообщения посылаются передатчиком повторно

Автоматы, моделирующие протокол PAR (Positive Acknowledge with Retransmissions)

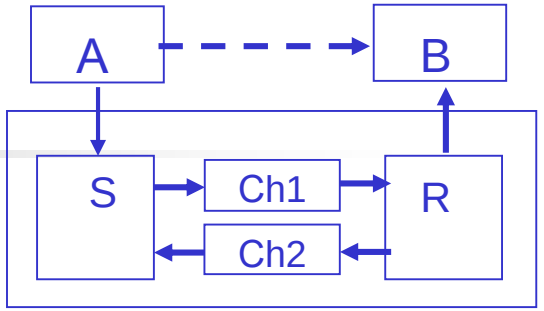


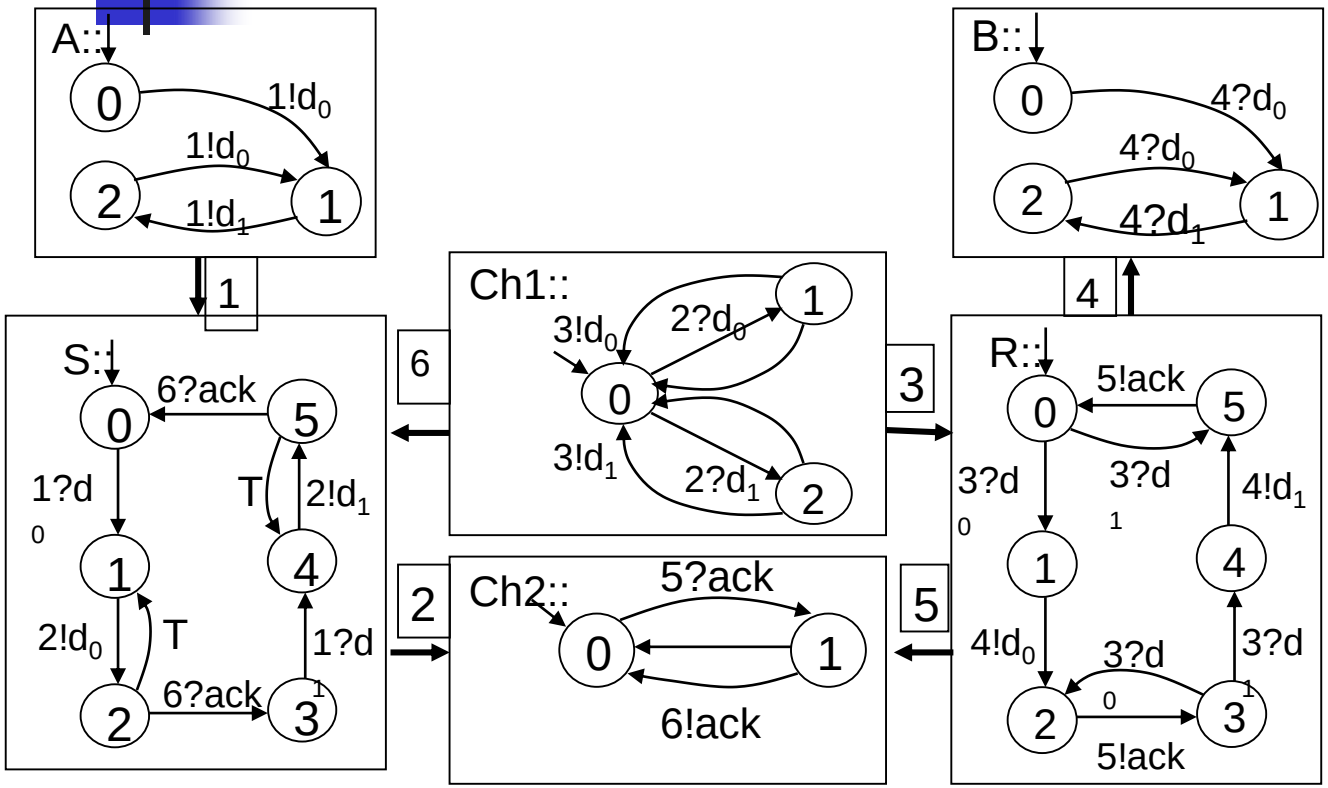
Определения

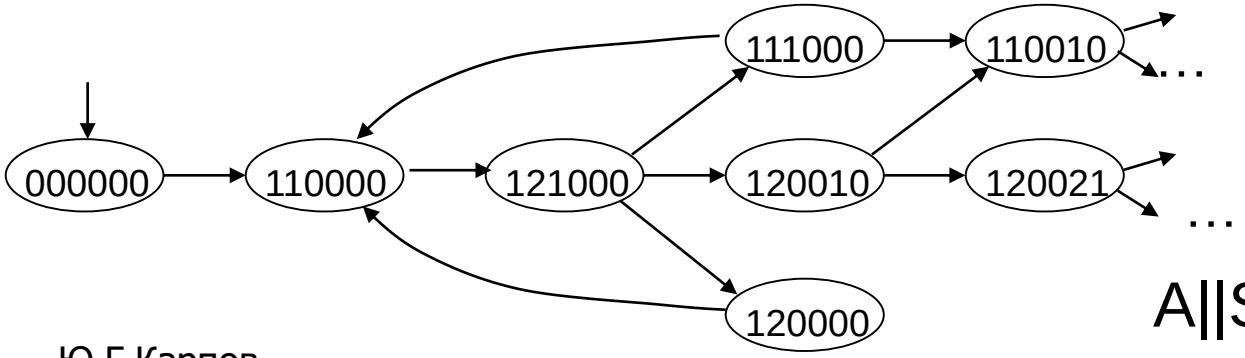
- Глобальное состояние - “snapshot” системы в момент времени ti.
- Вычисление – цепочка глобальных состояний, траектория.
- Поведение системы можно представить как дерево вычислений или множество вычислений



Система переходов протокола PAR



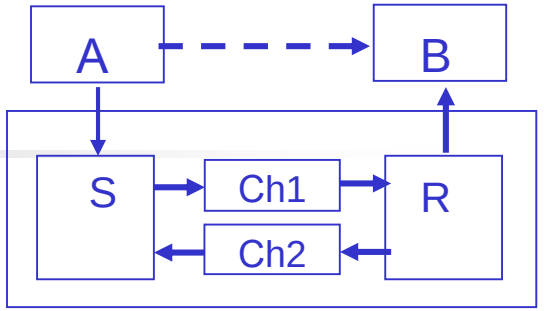




Дерево поведений

A||S||Ch1||Ch2||R||B

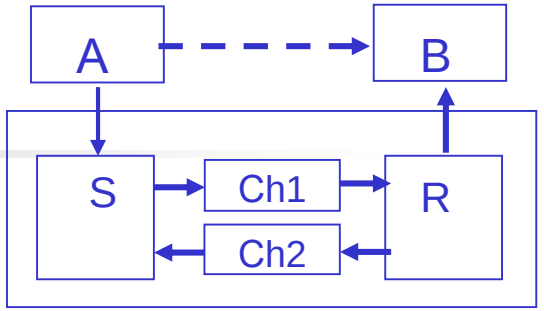
Проверка корректности протокола PAR



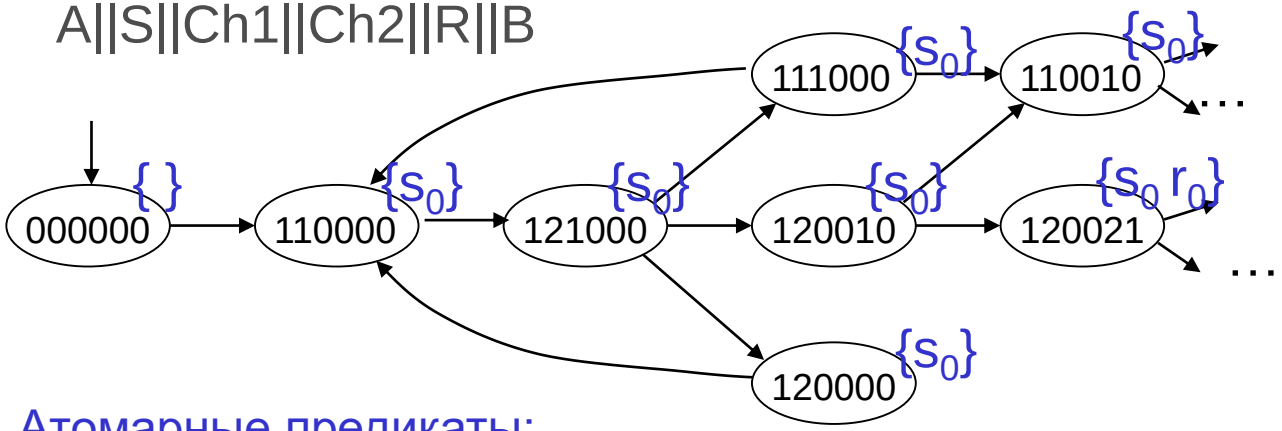
Spec1: “*если A послал сообщение, то когда-нибудь в будущем он сможет послать следующее сообщение*”

Spec 2: “*если A послал сообщение, то он не посылает следующее, пока B не примет его, и если B принял сообщение, то он не принимает следующее, пока A не пошлет его*”

Добавление атомарных предикатов. Структура Крипке протокола PAR



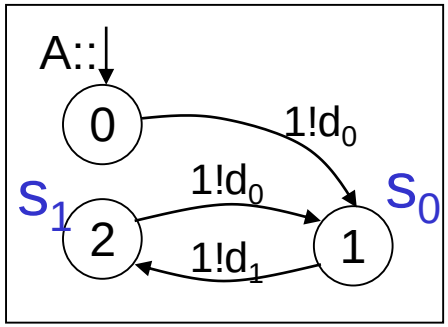
A||S||Ch1||Ch2||R||B

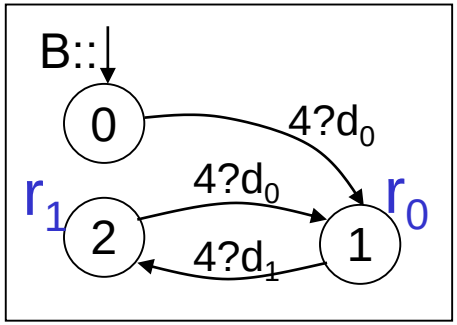


Атомарные предикаты:

s_k: *A послал сообщение, занумерованное k*

r_k: *B принял сообщение, занумерованное k*





s_0 *истинен в состоянии 1 автомата A*

s_1 *истинен в состоянии 2 автомата A*

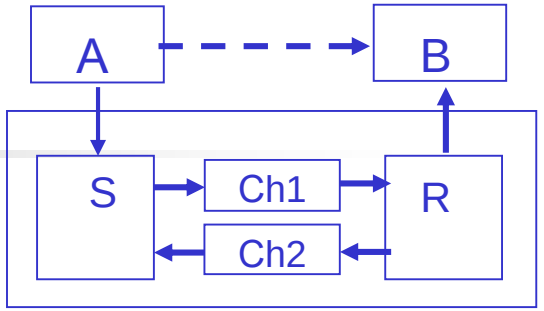
r_0 *истинен в состоянии 1 автомата B*

r_1 *истинен в состоянии 2 автомата B*

000000{ }→110000 {s_0}→121000 {s_0}→120010 {s_0} →110021 {s_0,r_0}→ …

{ } → {s_0} → {s_0} → {s_0} → {s_0,r_0} → …

Спецификация протокола PAR



Spec1: “*если A послал сообщение, то когда-нибудь в будущем он сможет послать следующее сообщение*”

$$\mathbf{G}((s_0 \rightarrow \mathbf{F}s_1) \wedge (s_1 \rightarrow \mathbf{F}s_0))$$

Spec 2: “*если A послал сообщение, то он не посылает следующее, пока B не примет его, и если B принял сообщение, то он не принимает следующее, пока A не пошлет его*”

$$\mathbf{G}(s_0 \rightarrow (\neg s_1 \mathbf{U} r_0)) \wedge \mathbf{G}(s_1 \rightarrow (\neg s_0 \mathbf{U} r_1)) \wedge \mathbf{G}(r_0 \rightarrow (\neg r_1 \mathbf{U} s_1)) \wedge \mathbf{G}(r_1 \rightarrow (\neg r_0 \mathbf{U} s_0))$$