Верификация параллельных программных и аппаратных систем

Курс лекций

Карпов Юрий Глебович
профессор, д.т.н., зав.кафедрой
“Распределенные вычисления и компьютерные сети”
Санкт-Петербургского политехнического университета

karpov@dcn.infos.ru

План курса

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

Лекция 5

Алгоритм model checking для проверки формул CTL

Model Checking

Система

Свойства системы

Спецификация системы (формальная модель)

Состояния и переходы (структура Крипке)

Спецификация требований (формальный язык)

Формула темпоральной логики

Model checking алгоритм – это проверка того, является ли данная структура Крипке МОДЕЛЬЮ формулы темпоральной логики

Процедура проверки

Model checking algorithm

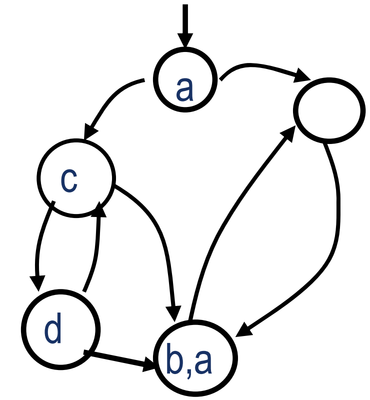
Структура Крипке – это модель проверяемой системы, поэтому термин Model checking часто понимают как ПРОВЕРКА МОДЕЛИ (системы)

Наша задача – рассмотреть алгоритм Model checking для логики CTL*

Расширенная логика ветвящегося времени CTL*

Темпоральные логики ветвящегося времени рассматривают возможные вычисления (пути на дереве) - траектории на развертке структуры Крипке

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

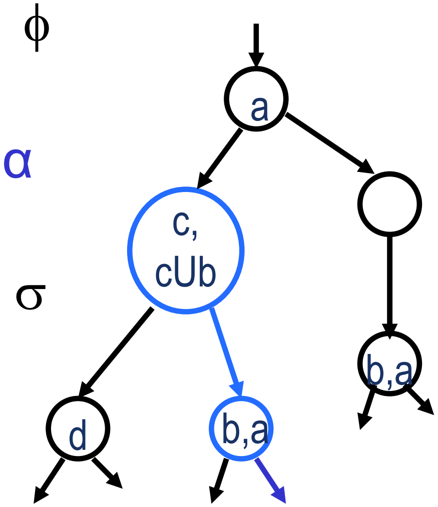


Грамматика. Формула CTL* - это формула состояний ϕ

- Формулы состояний $\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid E\,\alpha \mid A\alpha$
- Формулы путей $\alpha ::= \phi \mid \neg\alpha \mid \alpha\vee\alpha \mid \alpha U\alpha \mid X\alpha \mid G\alpha \mid F\alpha$

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

$A\,\alpha$, $G\alpha$, $F\alpha$ являются выводимыми формулами



Формула пути имеет смысл только если зафиксирован путь!

В состояниях могут стоять только формулы состояний!

LTL и CTL – подклассы CTL*

Формулы LTL:

$\mathbf{AG}(\, p \Rightarrow \mathbf{F}\ q\,)$

$\mathbf{A}\ (\neg a \vee \mathbf{G}b\ \&\ (a\mathbf{U}\ \neg c))$

$\mathbf{A}\ (\ a\ \mathbf{U}\ \neg b)$

Формулы CTL:

$\mathbf{AG}(\ p \& \neg \mathbf{EF}(q \Rightarrow r)\ )$

$\mathbf{EF}(\ a\ \&\ \mathbf{E}(a\mathbf{U}\ \neg c))$

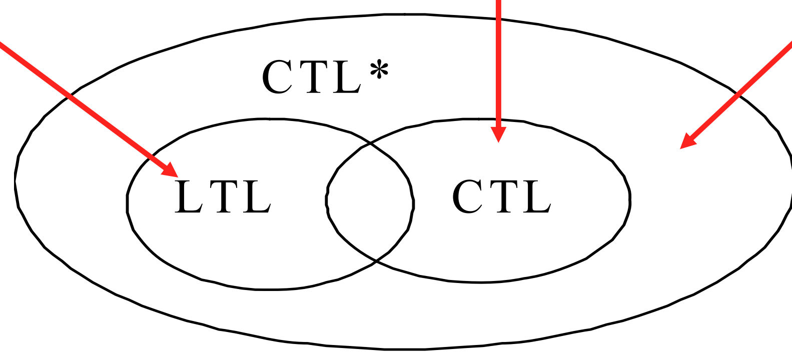
$\mathbf{A}\ (a\ \mathbf{U}\ \neg\ b\ )$

Формулы CTL*:

$\mathbf{E}(\neg p\ \&\ \mathbf{X}\ \mathbf{A}\ \mathbf{F}\ q)$

$\mathbf{EX}\ (a\ \&\ \mathbf{AX}(b\mathbf{U}c)\ ]$

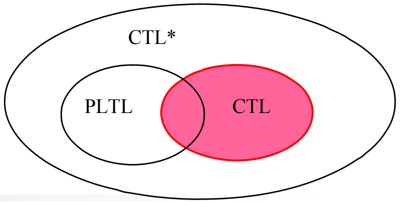
$\mathbf{A}\ (a\ \mathbf{U}\ \neg\ (\mathbf{F}\ b)\ )$



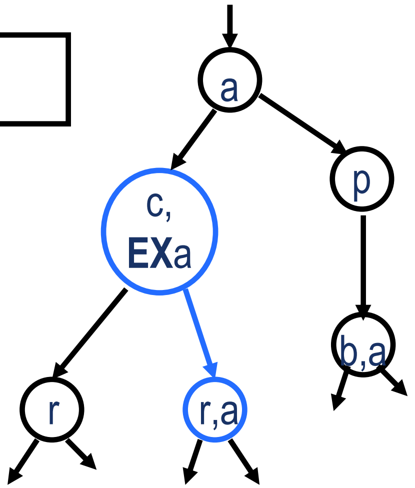
В LTL - формулы пути, которые должны выполняться для всех вычислений, т.е. предваряются квантором пути А

В CTL каждый темпоральный оператор предваряется квантором пути А или Е

CTL – подмножество CTL*



В CTL – только формулы состояний



Существует 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∨q)], **EX**p∧**EX**q, **EG AF** p, ...

Нельзя выразить **EGF**p, **A** [**X**p∨**XX**r], …

CTL оказалась чрезвычайно удобной вследствие эффективности алгоритма проверки выполнимости ее формул на структурах Крипке

Неформальное определение CTL

Синтаксис (грамматика):

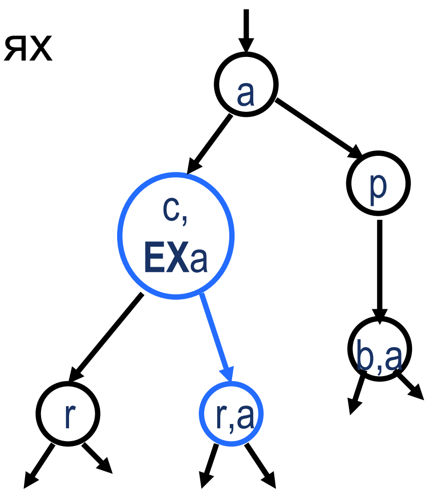
$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi_1 U \varphi_2] \mid E[\varphi_1 U \varphi_2]$

AXφ – формула φ выполняется во всех следующих состояниях

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



Все формулы CTL – это формулы состояний!!

Неформальное определение CTL (2)

Синтаксис (грамматика):

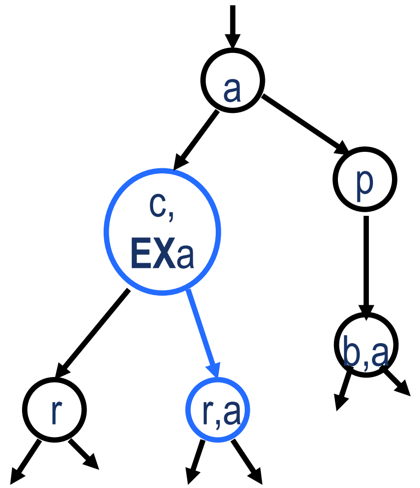
$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{AX}\varphi \mid \mathbf{EX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{AG}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{A}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{E}[\varphi_1 \mathbf{U} \varphi_2]$

AGφ - на всех путях из текущего состояния во всех состояниях этих путей формула φ выполняется

EGφ - существует путь из текущего состояния, во всех состояниях которого формула φ выполняется

A$(\varphi_1$**U**$\varphi_2)$ - на всех путях из текущего состояния когда-нибудь выполнится формула φ_2, а до этого во всех состояниях выполняется формула φ_1

E$(\varphi_1$**U**$\varphi_2)$ – из текущего состояния существует путь, на котором когда-нибудь выполнится φ_2, а до этого во всех состояниях этого пути выполняется φ_1



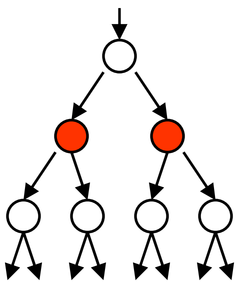
Все формулы CTL – это формулы состояний!!

CTL – рекурсивное определение

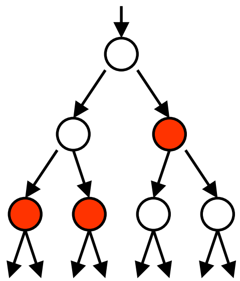
Синтаксис (грамматика):

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

AXred:

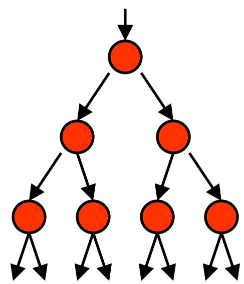


AFred:



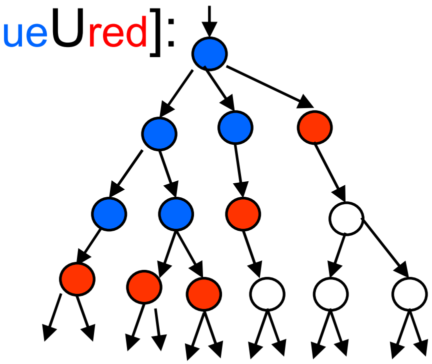
$AF\varphi = \varphi \vee AX\ AF\varphi$

AGred:



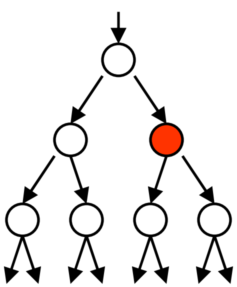
$AG\varphi = \varphi \wedge AX\ AG\varphi$

A [blueUred]:

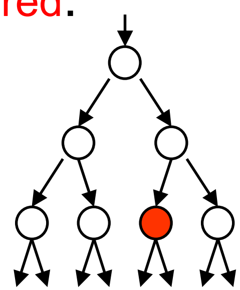


$A[\varphi U \phi] = \phi \vee \varphi \wedge AX\ A[\varphi U \phi]$

EXred:

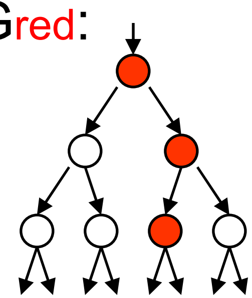


EFred:



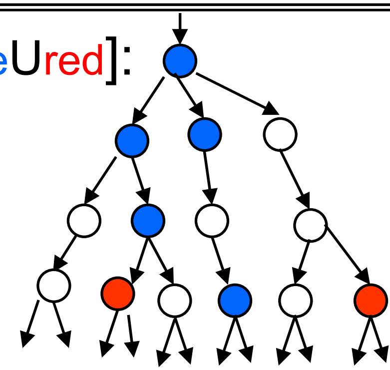
$EF\varphi = \varphi \vee EX\ EF\varphi$

EGred:



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

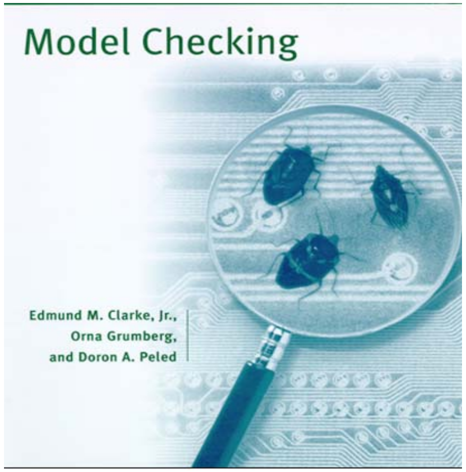
E [blueUred]:



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

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)



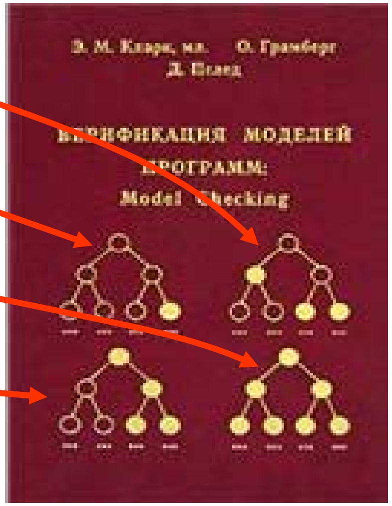
CTL формулы:

AF φ

EF φ

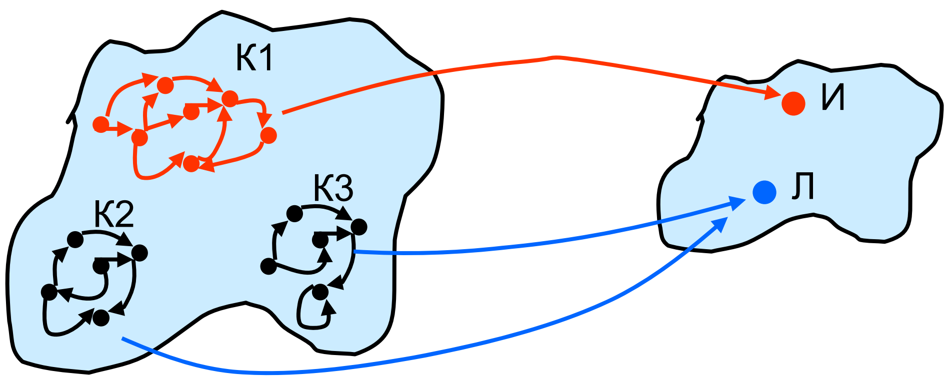
AG φ

EG φ



Model checking для CTL формул

$$\Phi = AG\,[\neg\, p \Rightarrow E(\,qUr)\,]$$

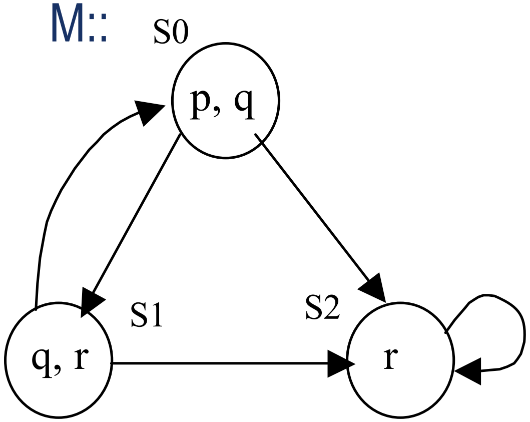


Model checking – это алгоритм проверки того, выполняется ли произвольная формула логики CTL на произвольной структуре Крипке, модели технической системы

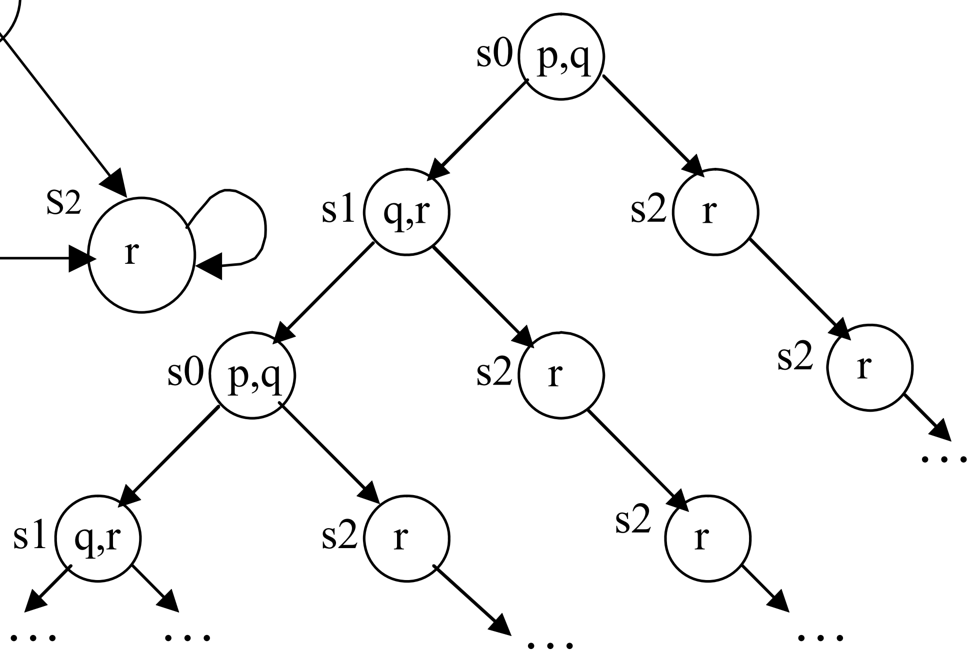
Как проверить выполнимость произвольной CTL формулы на произвольной структуре Крипке?

Model Checking для CTL – проверка на развертке (неформально)

M::



M::



1. $M, s_0 \models p \wedge q$
2. $M, s_0 \models \mathbf{EX}\ (q \wedge r)$
3. $M, s_0 \models \neg\ \mathbf{AX}(q \wedge r)$
4. $M, s_0 \models \neg\ \mathbf{EF}(p \wedge r)$
5. $M, s_0 \models \neg\ \mathbf{EG}r$
6. $M, s_0 \models\ \mathbf{AF}r$
7. $M, s_0 \models \mathbf{E}[\ (p \wedge q)\ \mathbf{U}\ r\ ]$
8. $M, s_0 \models\ \mathbf{A}[\ p\ \mathbf{U}\ r\ ]$
9. $M, s_0 \models\ \mathbf{EF}\ \mathbf{AG}r$

Этот анализ на бесконечных вычислениях неформален

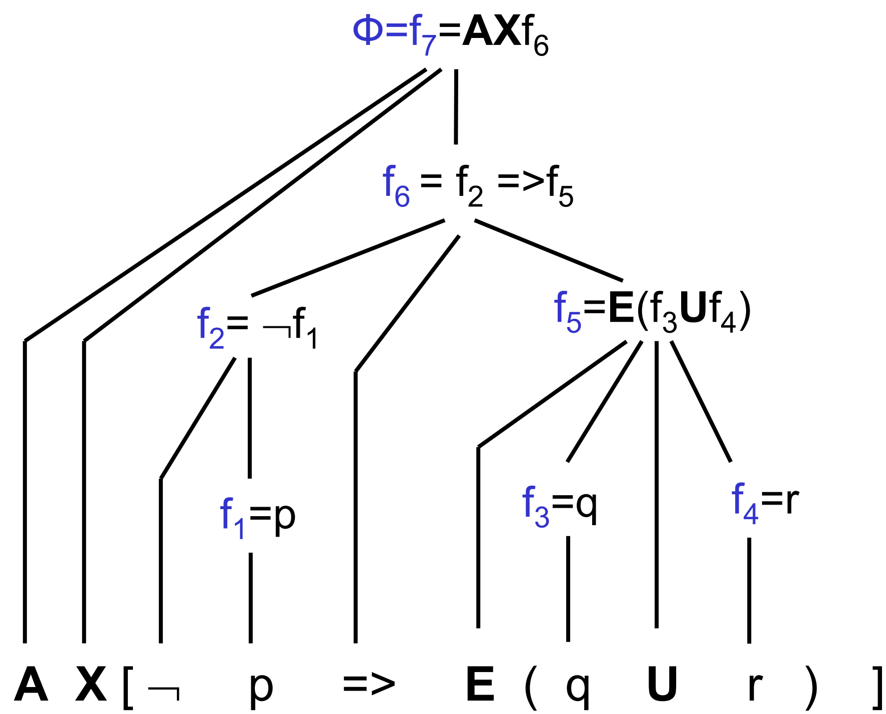
Необходимо разработать алгоритм проверки выполнимости любой темпоральной формулы на структуре Крипке

Алгоритм маркировки для CTL формул

Формул CTL – бесконечное число!!

Как проверить выполнимость произвольной формулы, например

Φ = **AX** [¬ p=> **E**(q**U**r)]?



Общая идея:

1) Формул – бесконечное число, но число типов подформул – конечно!

2) Помечаем (маркируем) все состояния структуры Крипке подформулами формулы Φ, которые истинны в этих состояниях

3) Если начальное состояние структуры Крипке M помечено Φ, то Φ выполняется на M

Вывод: необходимо разработать алгоритмы маркировки состояний структуры Крипке для каждой возможной подформулы формулы Φ

Подформулы CTL формул

Синтаксис (грамматика):

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

Нужно разработать алгоритмы маркировки для всех возможных подформул CTL формул. Все возможные подформулы задаются грамматикой

Эти алгоритмы для каждого типа формул должны определять, выполняется ли формула φ в каждом состоянии структуры Крипке в том предположении, что выполнение или невыполнение подформул φ уже определено в состояниях

Пусть структура Крипке M задана. Обозначим: $s \models \phi$ - в состоянии s структуры Крипке M формула ϕ выполняется. Тогда:

$s \models p$ 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

$s \models EX\varphi$ если и только если существует путь из s, в следующем состоянии которого выполняется φ

Это, фактически, определение семантики формул CTL

Формальная семантика CTL

Синтаксис (грамматика):

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

$s \models p \quad \equiv \quad p \in L(s);$

$s \models \neg\varphi \quad \equiv \quad s \not\models \varphi;$

$s \models \varphi_1 \vee \varphi_2 \equiv s \models \varphi_1 \vee s \models \varphi_2;$

$s \models AX\varphi \equiv (\forall s_1: s \rightarrow s_1)\ s_1 \models \varphi;$

$s \models EX\varphi \equiv (\exists s_1: s \rightarrow s_1)\ s_1 \models \varphi;$

$s \models AG\varphi \equiv \forall(s_0 \rightarrow s_1 \rightarrow \ldots)\ (s=s_0)\ \ (\forall i)\ s_i \models \varphi;$

$s \models EG\varphi \equiv \exists(s_0 \rightarrow s_1 \rightarrow \ldots)\ (s=s_0)\ \ (\forall i)\ s_i \models \varphi;$

$s \models AF\varphi \equiv \forall(s_0 \rightarrow s_1 \rightarrow \ldots)\ (s=s_0)\ \ (\exists i)\ s_i \models \varphi;$

$s \models EF\varphi \equiv \exists(s_0 \rightarrow s_1 \rightarrow \ldots)\ (s=s_0)\ \ (\exists i)\ s_i \models \varphi;$

$s \models A(\varphi_1 U \varphi_2) \equiv \forall(s_0 \rightarrow s_1 \rightarrow \ldots)\ (s=s_0)\ \ (\exists j: 0 \le j)\ (s_j \models \varphi_2 \wedge (\forall k: 0 \le k < j)\ s_k \models \varphi_1);$

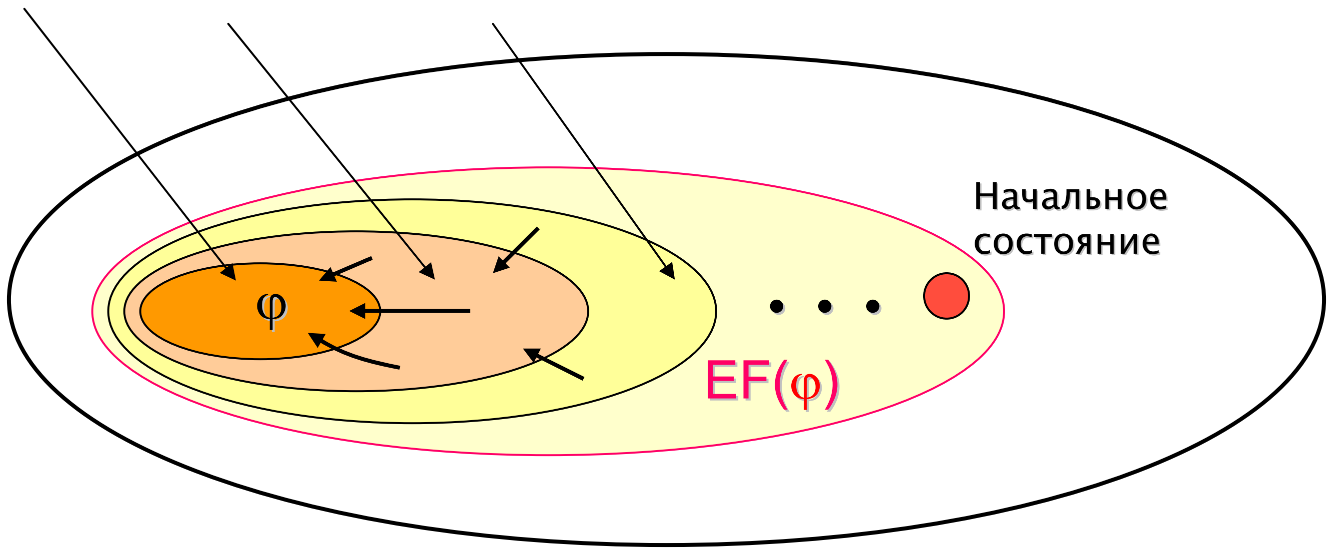
$s \models E(\varphi_1 U \varphi_2) \equiv \exists(s_0 \rightarrow s_1 \rightarrow \ldots)\ (s=s_0)\ \ (\exists j: 0 \le j)\ (s_j \models \varphi_2 \wedge (\forall k: 0 \le k < j)\ s_k \models \varphi_1).$

Определение множества состояний, удовлетворяющих формуле EFφ

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

EF(φ) ≡ состояния из которых можно достичь φ - это состояния, помеченные:

$\varphi \ \cup \ EX(\varphi) \ \cup \ EX(EX(\varphi)) \cup \ldots$

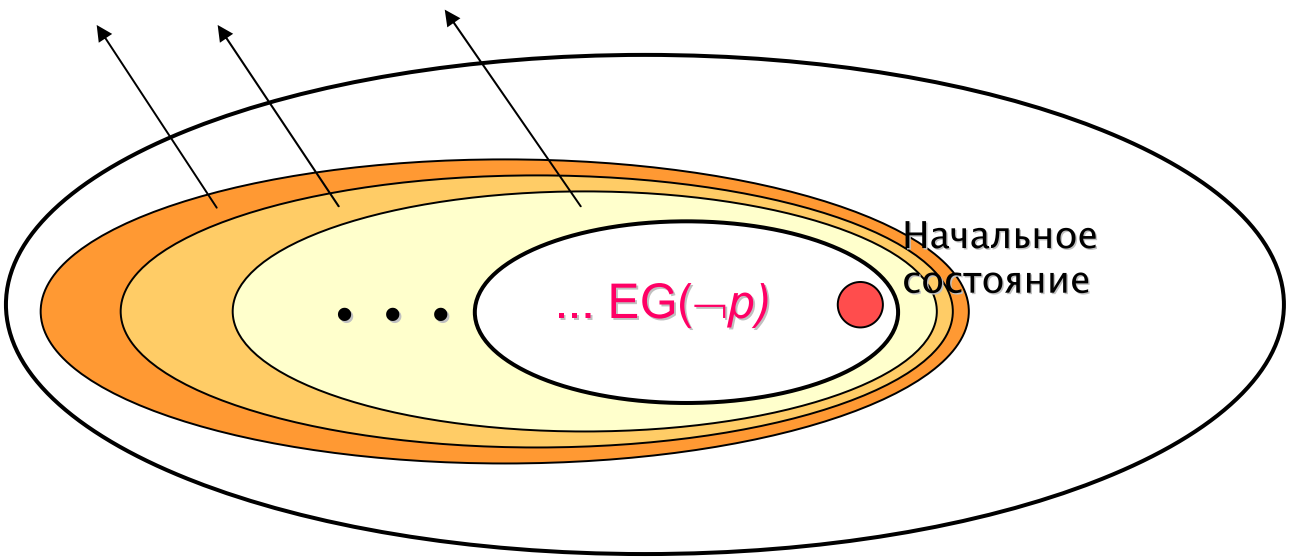


Если в начальном состоянии выполняется EFφ, то структура Крипке удовлетворяет EFφ

Определение множества состояний, удовлетворяющих формуле EGφ

EGφ = φ ∧ EXφ ∧ EX EXφ ∧EX EX EXφ ∧ ...

EG(φ) ≡ состояния на которых выполняется φ, и из которых можно достичь φ - это состояния:
φ ∩ EX(φ) ∩ EX(EX(φ)) ∩ ...



Если в начальном состоянии выполняется EGφ, то структура Крипке удовлетворяет EGφ

Базисы CTL

Синтаксис (грамматика):

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

Но нужны ли все эти конструкции?

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

2. Используем следующее соотношение:
$F\varphi \equiv \neg G\neg\varphi$. Отсюда, $AG\varphi \equiv \neg EF\neg\varphi$, т.е. AG можно заменить на EF, и наоборот. AG и EF называются дуальными, взаимозаменяемыми.

3. Аналогично, EG и AF – дуальны, взаимозаменяемы

4. Очевидно $AX\varphi \equiv \neg E\neg X\varphi$. Но очевидно, что $\neg X\varphi \equiv X\neg\varphi$. Поэтому $AX\varphi \equiv \neg EX\neg\varphi$
Отсюда: AX и EX – дуальны, взаимозаменяемы

5. Известно: $F\varphi \equiv$ *True*$U\varphi$, поэтому EF можно заменить на EU**,** а AF – на AU

6. Оказывается: $A(\varphi_1 U \varphi_2) \equiv AF\varphi_2 \wedge \neg E(\neg \varphi_2 U(\neg\varphi_1 \wedge \neg\varphi_2))$

Взаимозависимости комбинаторов CTL

Взаимозависимости CTL формул

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

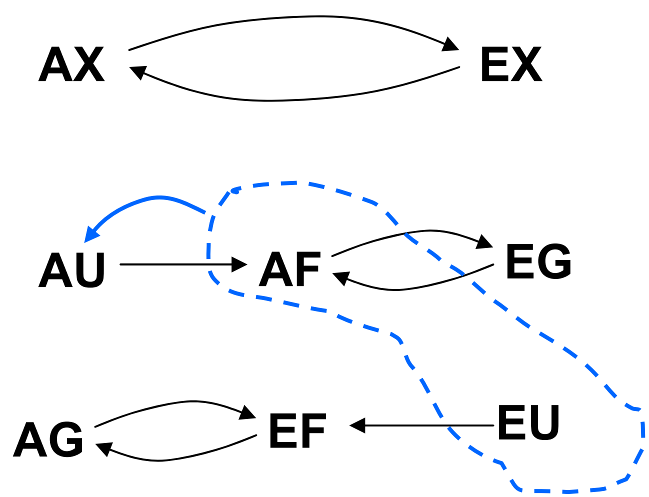
$\mathbf{EG}\varphi \equiv \neg\mathbf{AF}\neg\varphi$

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

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

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

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



Возможные базисы CTL

{ **EX**, **AF**, **EU** },
{ **AX**, **AU**, **EU** },
{ **EX**, **EG**, **EU** } и т.д.

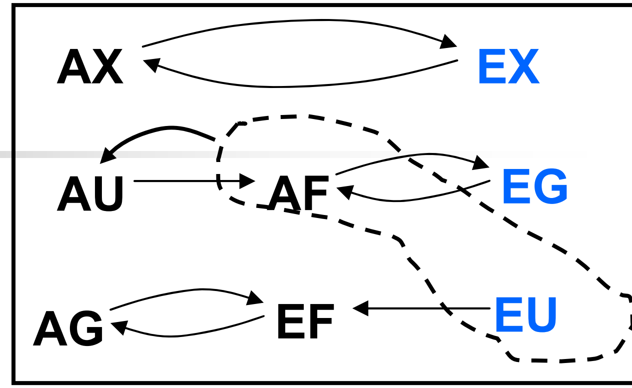
Всего существует 6 базисов CTL

Для процедуры Model checking достаточно построить алгоритмы маркировки только для CTL комбинаторов какого-нибудь базиса

Базисы CTL

Синтаксис (грамматика):

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



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

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

Sat_Ψ - множество состояний структуры Крипке, в которых выполняется формула Ψ

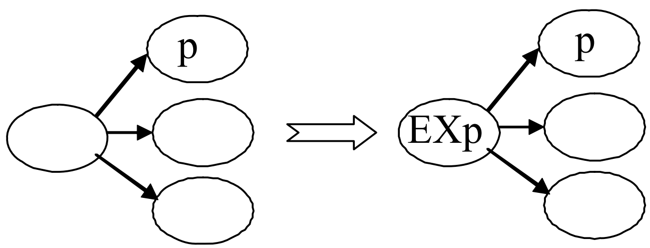
```
for all 0<i≤|Φ| do
  for all Ψ ∈ Sub(Φ) with |Ψ| = i do
    switch(Ψ):
        true     :      SatΨ := S;
        p        :      SatΨ := { s∈S| p∈L(s) };
        ¬p       :      SatΨ := { s∈S| p∉L(s) };
        p∧q      :      SatΨ := { s∈S| p,q ∈L(s) };
        EXp      :      SatΨ := Sat_EX(p);
        AFp      :      SatΨ := Sat_AF(p);
        E(pUq)  :      SatΨ := Sat_EU(p,q);
    end switch
  /* Sat_EX, Sat_AF и Sat_EU- это функции, определенные далее     */
  /* Все состояния из SatΨ помечаем новым атомарным предикатом pΨ:*/
  for all s∈ SatΨ do L(s):=L(s)∪{pΨ } od
od
```

Этот алгоритм базируется на парсере – алгоритме синтаксического анализа формулы

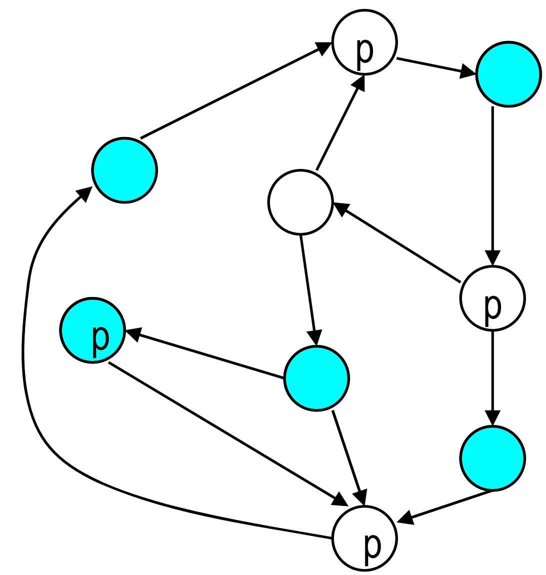
Основная идея – для каждой подформулы определить, в каких состояниях структуры Крипке М эта подформула выполняется. Если φ выполняется в НАЧАЛЬНОМ состоянии М, то мы считаем, что она выполняется на М

Алгоритм Model Checking для CTL (EX)

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




```
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
```



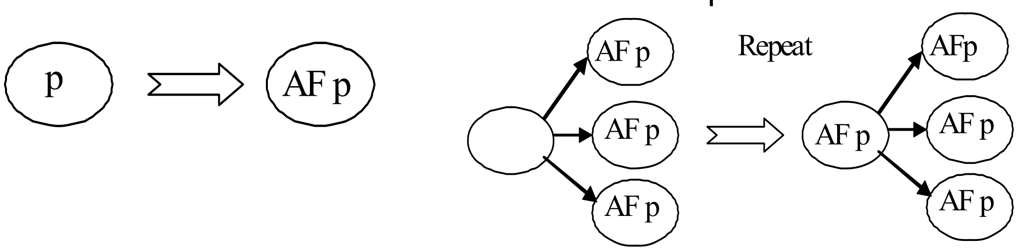
Все закрашенные состояния попадают в множество SAT_EX(p)

Подобные алгоритмы легко можно построить и для AX, EF, AU

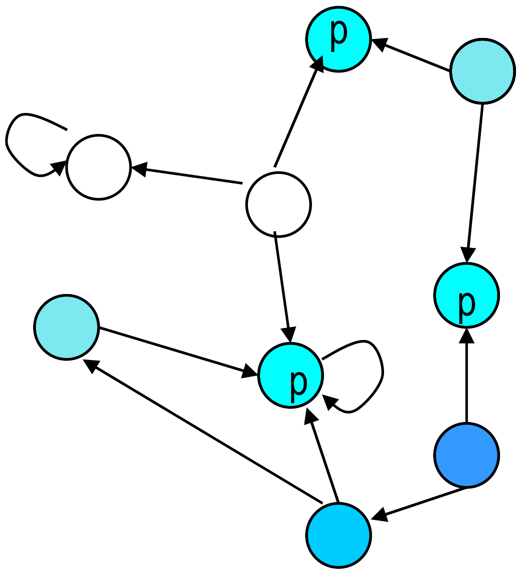
Алгоритм для EG строится по-другому. Алгоритм для AG обычно не используется

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}



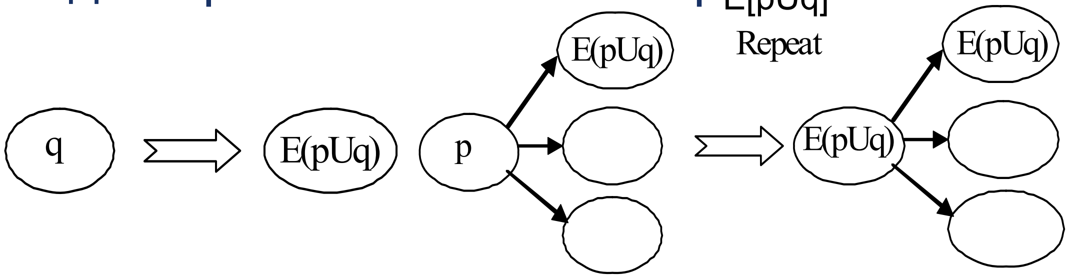

```
function SAT_AF(p)
                  /*дает все s, в которых истинна AFp */
    local var X,Y
    begin
        X:=S;
        Y:= SAT (p);
        repeat until X=Y
        begin
          X:=Y;
          Y:= Y ∪ { s | (∀s1:s→s1) s1∈Y }
         end
         return Y
     end
```



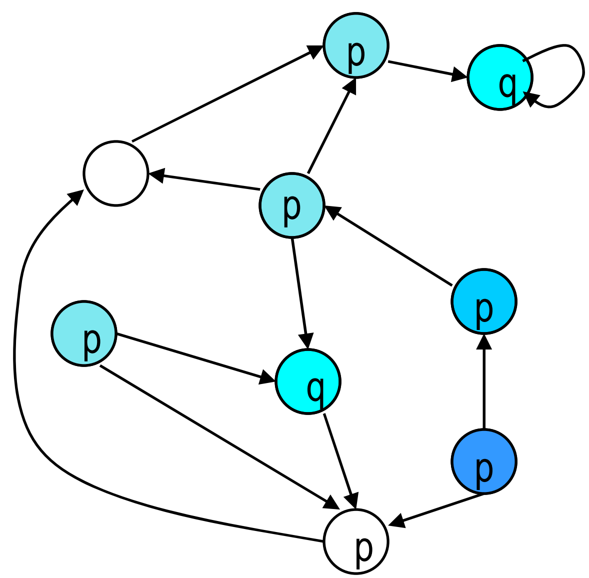
Все закрашенные состояния попадают в множество $SAT_{AF}(p)$

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 $p_{E[pUq]}$



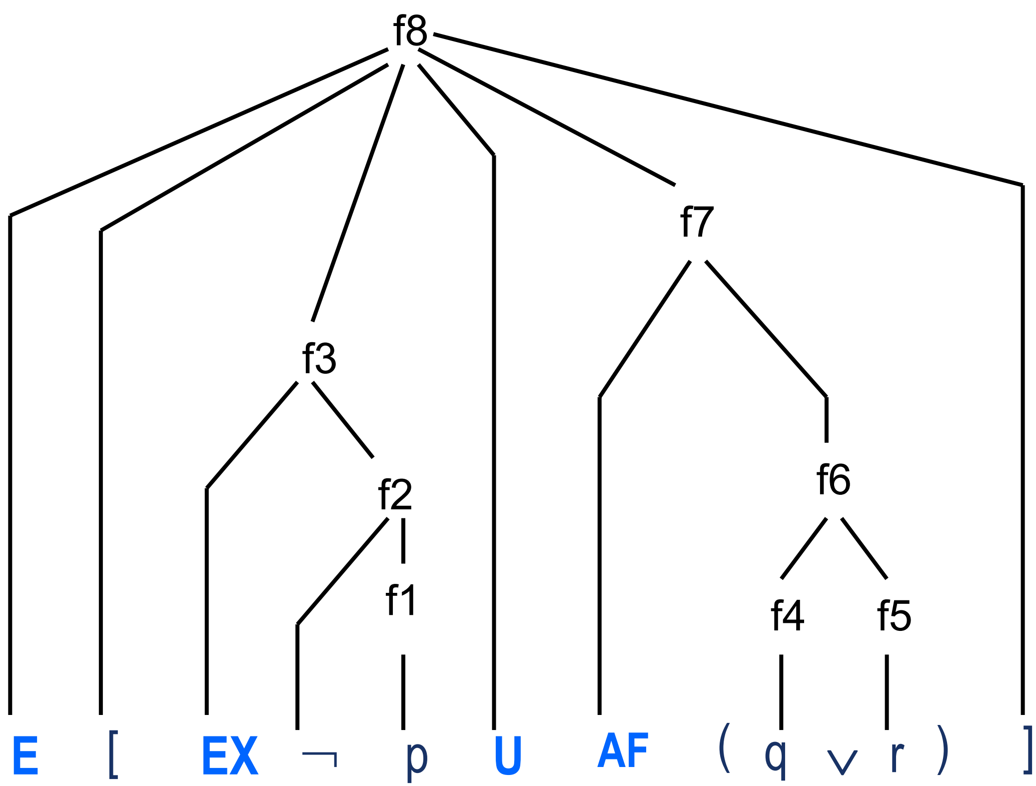

```
function SAT_EU (p, q) /*дает все s, удовл E(pUq) */
    local var P, X, Y
    begin
        P:= SAT (p);  X:=S;  Y:= SAT (q);
        repeat until X=Y
        begin
          X:=Y;
          Y:= Y ∪ (P ∩ { s | (∃s₁∈Y) s→s₁ } )
         end
         return Y
    end
```



Все закрашенные состояния попадают в множество SAT_EU(p,q)

Model Checking: синтаксический анализ формулы

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



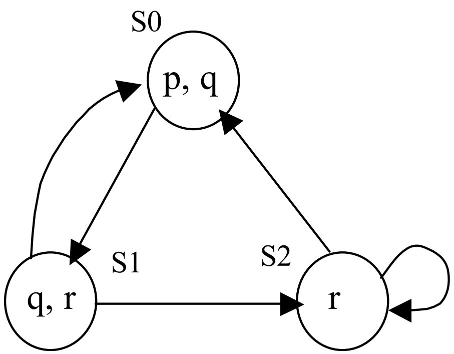
Последовательно снизу вверх:

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

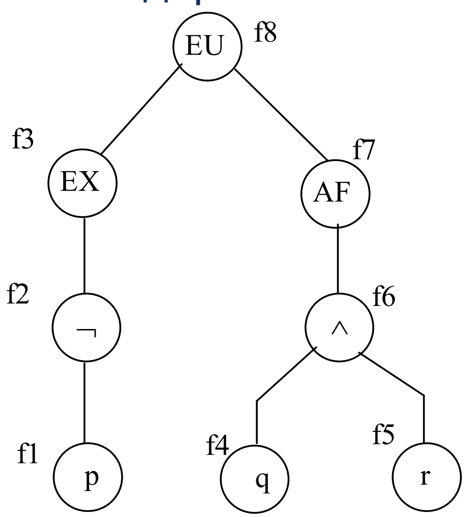
Обычный синтаксический анализ

Model Checking – пример

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



Синтаксическое дерево:



$f1 = p$

$f2 = \neg f1$

$f3 = EX\ f2$

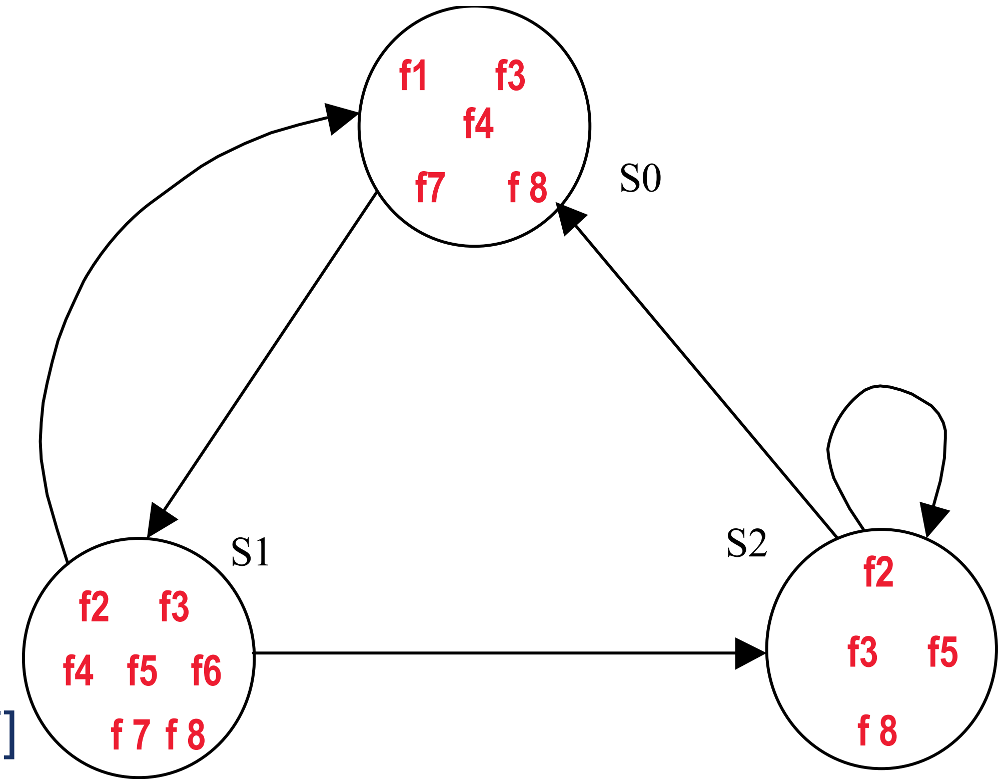
$f4 = q$

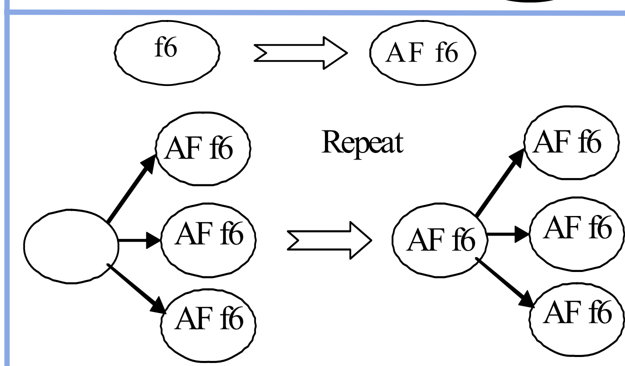
$f5 = r$

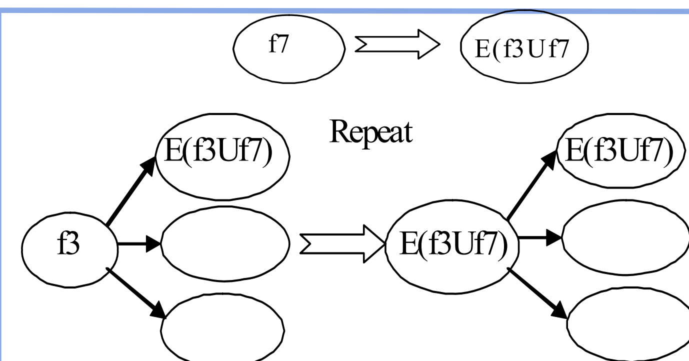
$f6 = f4 \wedge f5$

$f7 = AF\ f6$

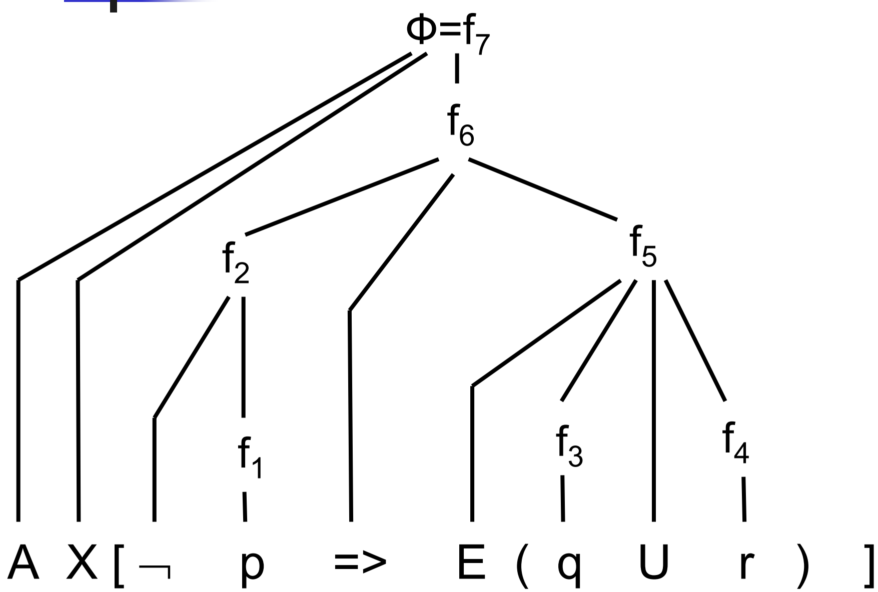
$f8 = E[\ f3\ U\ f7]$

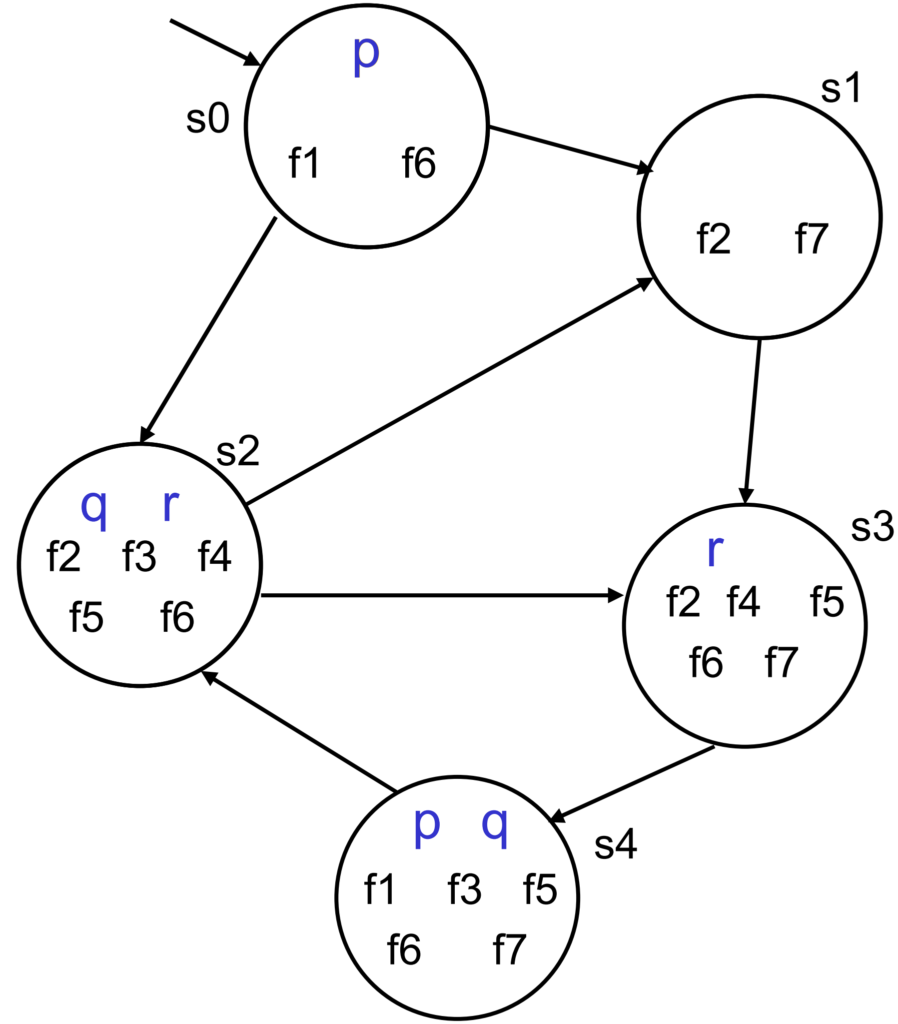






Алгоритм маркировки для CTL формул





$f_1 = p$ — f_1 : { s0, s4 }

$f_2 = \neg f_1$ — f_2 : { s1, s2, s3 }

$f_3 = q$ — f_3 : { s2, s4 }

$f_4 = r$ — f_4 : { s2, s3 }

f5 = E(f3 U f4) — f5 : { s2, s3, s4 }

f6 = f2 $\Rightarrow$ f5 — f6 : { s0, s2, s3, s4 }

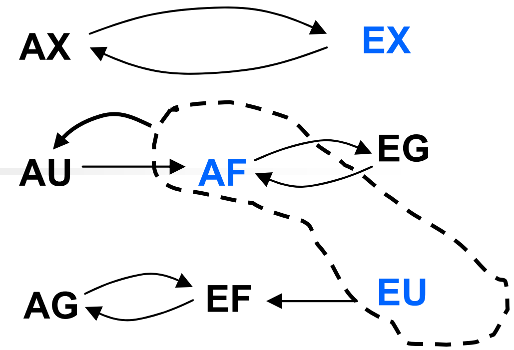
f7 = Φ = AX f6 — f7 : { s1, s3, s4 }

На M не выполняется Φ

Разработаны инструменты верификации

- SPIN - Bell Labs – (|| взаимодействующие процессы) LTL
  - http://cm.bell-labs.com/cm/cs/what/spin/
- UPPAAL - Uppsala University, Швеция (системы РВ) CTL+Time
  - http://www.docs.uu.se/docs/rtmv/uppaal/
- VIS - Uni Berkeley, Uni Colorado
  - http://vlsi.colorado.edu/~vis/
- KRONOS – INRIA (системы реального времени)
  - http://www.inrialpes.fr/vasy/cadp/software/99-c-kronos.html
- HYTECH - Cornell University (линейные гибридные системы)
  - http://www.henzinger.com/monika/hytech.html
- SMV – Carnegie Mellon University (Symbolic Model Verificator)
  - http://www.cs.cmu.edu/~modelcheck/smv.html CTL Symbolic
- STeP – Stanford University (Stanford Temporal Prover)
  - http://www-step.stanford.edu/

Резюме: Алгоритм Model Checking для CTL



Вход: Структура Крипке M=(S, $\rightarrow$, L) и CTL формула Ф

Выход: Множество состояний, удовлетворяющих формуле Ф

1 шаг: Трансляция Ф в базис {$\neg$, $\vee$, AF, EU, EX} (можно и в любой другой!)

2 шаг: По формуле Ф строятся все ее подформулы (синтаксический анализ)

3 шаг: Все состояния M последовательно, помечаем новыми атомарными предикатами, определенными для подформул Ф, начиная с внутренних, которые в этих состояниях истинны

Если эта подформула:

p : то помечаем каждое состояние s атомом p, если $p \in L(s)$

$\neg$p : то помечаем каждое s новым атомарным предикатом $p_{\neg p}$, если s не помечена p

$p \vee q$: то помечаем каждое s новым атомарным предикатом $p_{p \vee q}$ если s помечена p или q

AFp: то помечаем каждое s новым атомом p_{AFp}, если оно помечено p;
повторяем: помечаем каждое s атомом p_{AFp}, если все преемники s помечены p_{AFp}

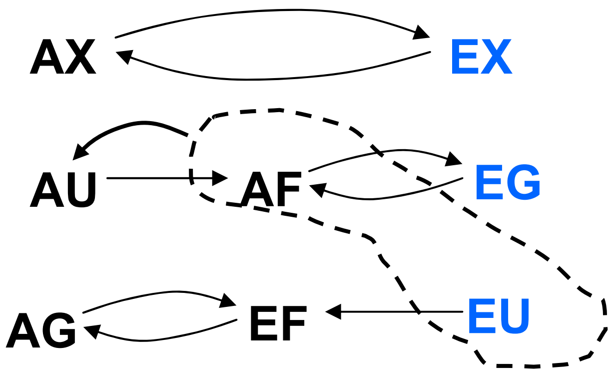
E[pUq] : то помечаем любое s новым атомом $p_{E[pUq]}$, если оно помечено q; повторяем: помечаем каждое s атомом $p_{E[pUq]}$, если s помечено p и хотя бы один преемник s помечен $p_{E[pUq]}$

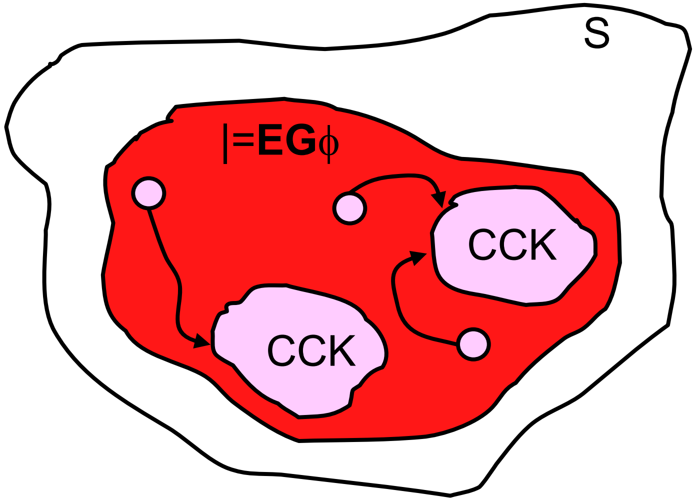
EXp : то помечаем каждое s атомом p_{EXp}, если хотя бы один из преемников s помечен p

Сложность алгоритма O[|Ф|*| S|*(|S| + | T|)]. Наиболее сложный алгоритм - для AF

Model Checking алгоритм (более эффективный)

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





EGϕ:

1. Выбрасываем из М все состояния, которые не помечены ϕ (и их переходы)

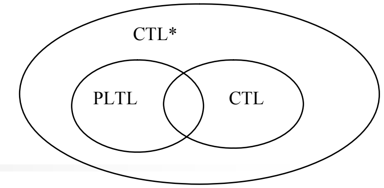
2. Находим все сильно связные компоненты (ССК) в оставшемся графе

3.Breadth-First обратный поиск всех состояний, которые связаны с любой ССК

Сложность алгоритма O[|Ф| *(| К |)]

Линеен как по размеру формулы Ф, так и по размеру модели (сумме состояний и переходов)

Сравнение логик LTL, CTL и CTL* (семантика)

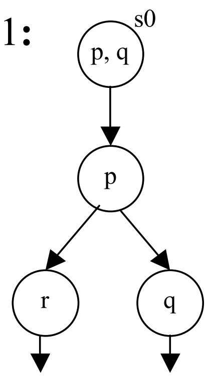


E[pU(qUr)] - не CTL-формула, но: E[pU(qUr)] = E[pUE(qUr)]

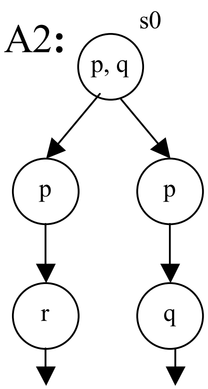
Выполняется ли соотношение между темпоральными логиками и семантически?

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

A1:



A2:



Доказательство. В PLTL поведения A1 и A2 неразличимы, 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:

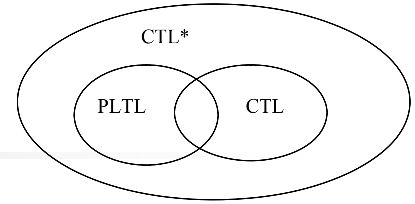
A1, s0 |= AX(EXq & EXr),

A2, s0 |≠ AX(EXq & EXr)

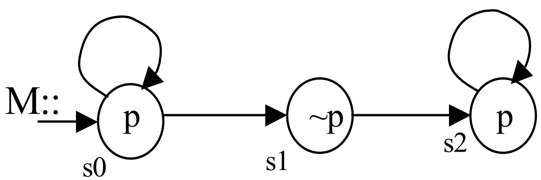
Теорема. Существуют свойства поведений, выражаемые в PLTL и не выражаемые в CTL

Доказательство. Свойство AGFp не выражается в CTL (доказано elsewhere)

Пример: свойство CTL*, не выражаемое в CTL



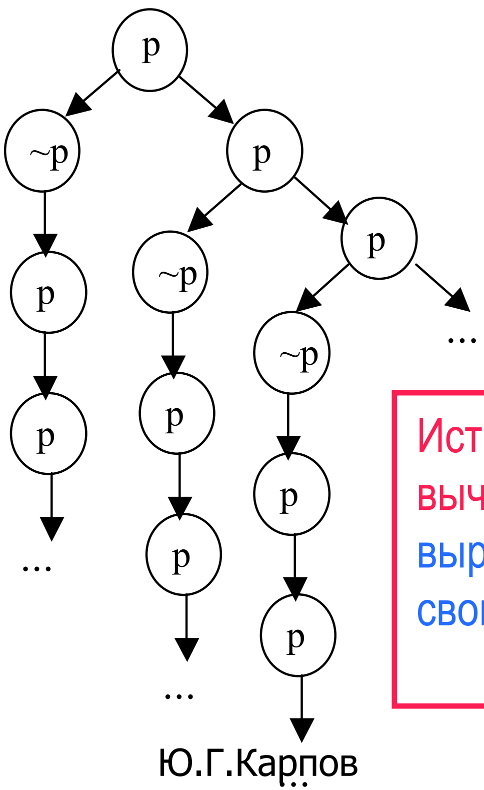
Теорема. Существуют свойства поведений, выражаемые в CTL* и не выражаемые в CTL



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 |$\neq$ AF AGp, поскольку та траектория, которая всегда остается в s0, может в любой момент перейти в состояние s1, в котором $\neg$p

Следовательно, AFGp $\neq$ AFAGp

Истинность CTL формул зависит от текущего состояния, но не от текущего вычисления (понятия текущего вычисления в CTL нет). CTL позволяет выразить свойство достижимости по пути, но не позволяет выразить другие свойства, которые могут встретиться вдоль пути

Заключение

- Множество интерпретаций формул TL– это множество всех возможных структур Крипке. Структура Крипке, на которой удовлетворяется формула TL, называется моделью этой формулы
- Проверка осуществляется последовательным вычислением истинности подформул ϕ для всех состояний структуры Крипке. Подформулы выделяются алгоритмом синтаксического анализа
- Существует несколько базисов для формул CTL
- Сложность алгоритма проверки того, является ли структура Крипке К моделью CTL-формулы Ф, удивительно мала: O(|K| *|Ф|)
- Последнее время наблюдается большая активность исследований по расширению этого подхода для временных и вероятностных моделей

Спасибо за внимание