Верификация параллельных программных и аппаратных систем

Курс лекций

Шошмина Ирина Владимировна

Карпов Юрий Глебович

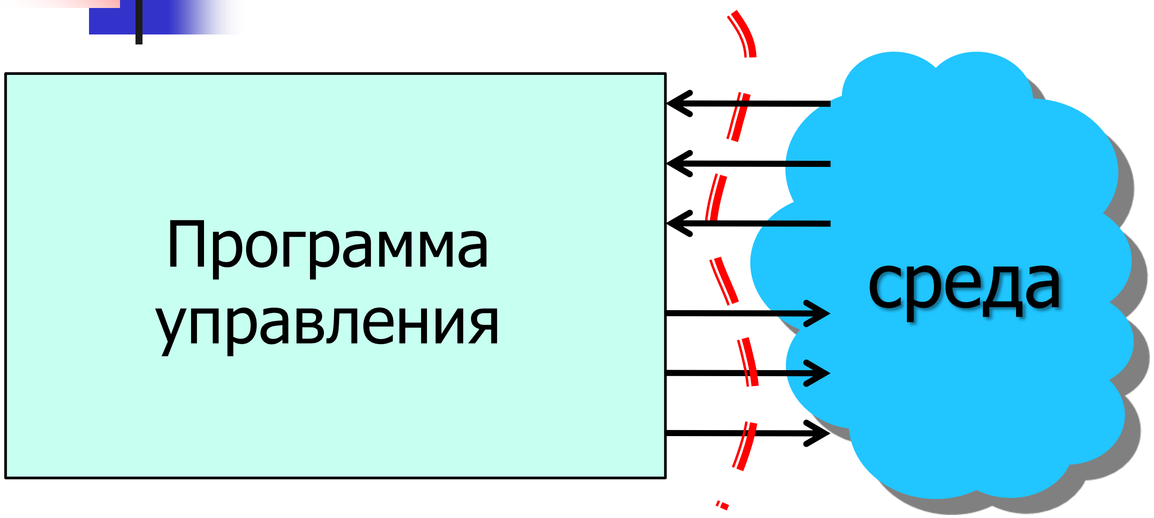
Лекция 8

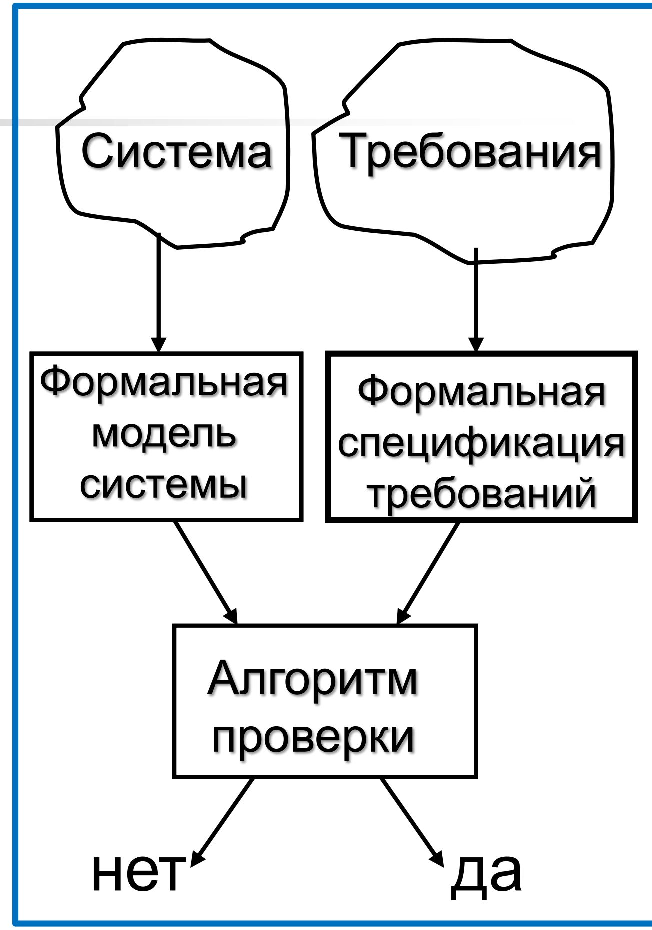
Model checking для формул LTL – часть 2

План курса

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

Объект анализа – другой. Это программы управления



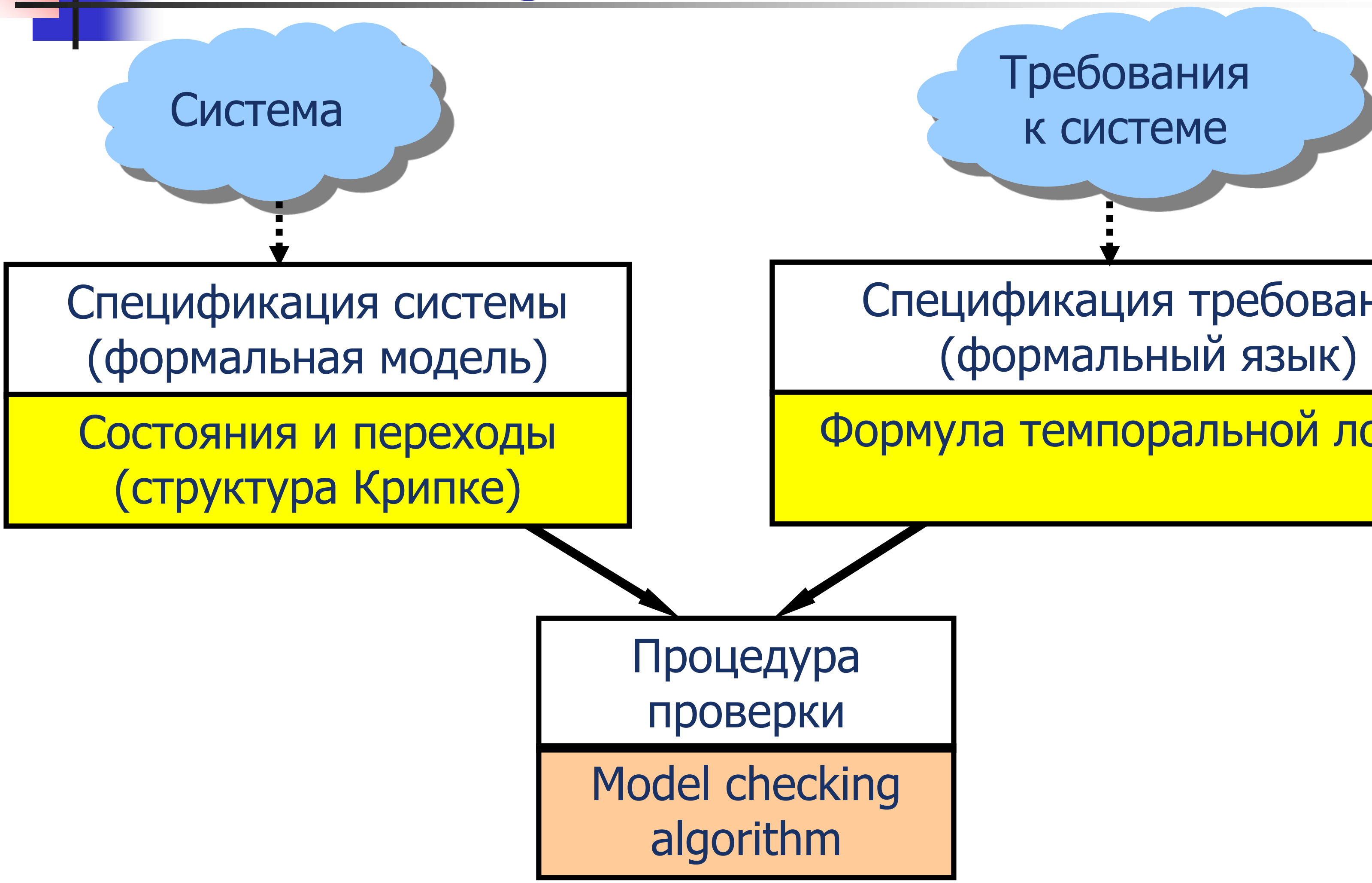


Программы управления НЕ останавливаются, они реагируют на **события** на интерфейсе.

Цель программы управления – поддержание определенного ШАБЛОНА взаимодействия на интерфейсе со средой.

Важен не конечный результат, а ***ПОВЕДЕНИЕ***

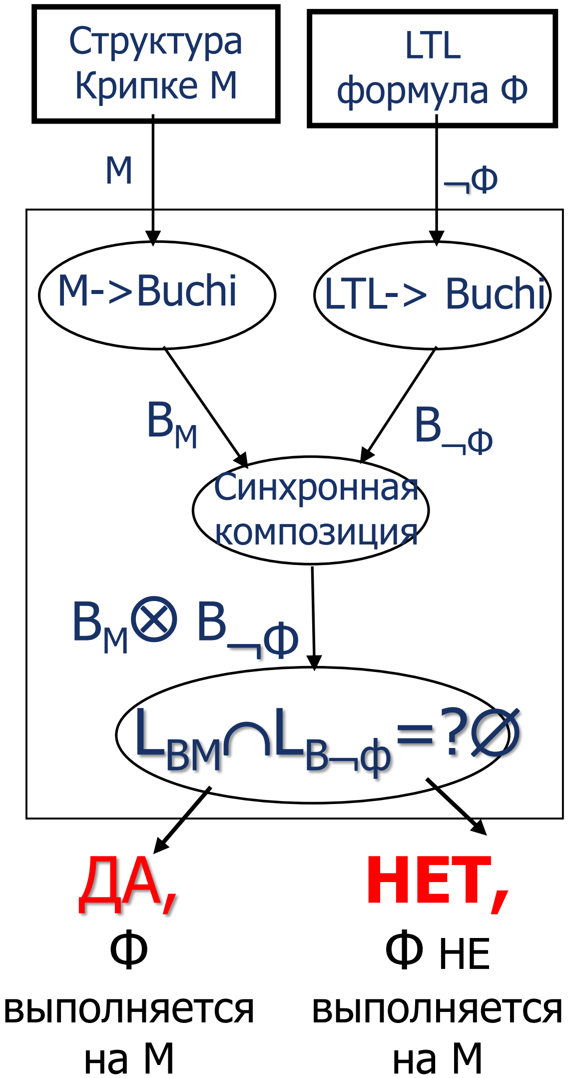
Model Checking



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

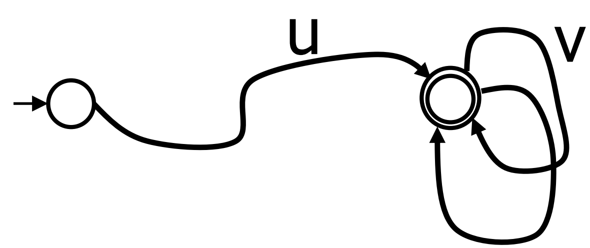
Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

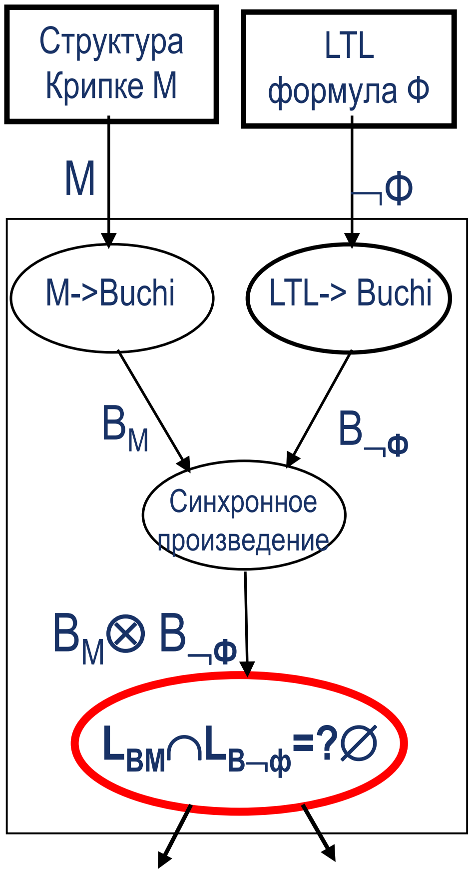
- 1. Построение по структуре Крипке M такого автомата Бюхи B_M, который допускает все возможные вычисления структуры M.
- 2. Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Ф автомата Бюхи $B_Ф$
- 3. Синхронная композиция двух автоматов Бюхи.
- 4. **Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи**.



Проблема пустоты автомата Бюхи

- σ допускается автоматом Бюхи A, iff существует заключительное состояние A, которое проходится бесконечное число раз при приеме ω-цепочки σ.
- Для того, чтобы автомат Бюхи B допускал хотя бы одну БЕСКОНЕЧНУЮ цепочку, в графе переходов B должен быть достижимый из начального состояния **ЦИКЛ**, включающий хотя бы одно из заключительных состояний.

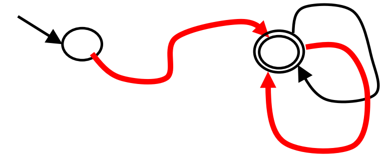




ω - язык, допускаемый автоматом Бюхи, НЕ пуст, если из начального состояния существует путь u в одно из принимающих состояний, и цикл v из этого состояния в себя.

Th. Проблема пустоты языка, допускаемого автоматом Бюхи, разрешима.

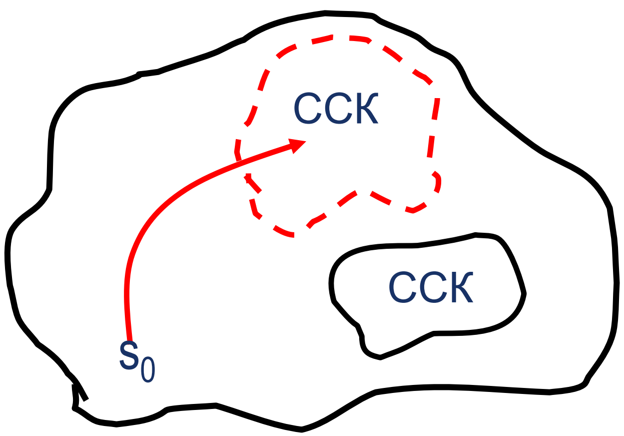
Контрпример: цепочка, переводящая автомат в цикл принимающих состояний

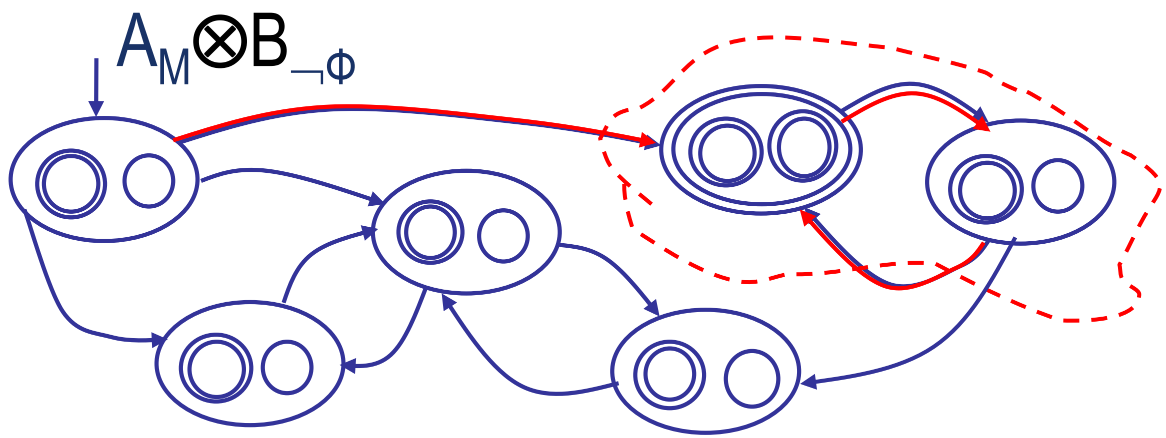


1. Находим все Сильно Связные Компоненты графа переходов.
2. Оставляем те ССК, в которых есть хотя бы одно допускающее состояние.
3. Проверяем, есть ли путь из S_0 хотя бы в одну оставшуюся ССК.

Алгоритм нахождения ССК линеен, проверка достижимости каждой компоненты также требует линейного времени $\Rightarrow$ Сложность $O(n^2)$.

Каждая цепочка из начального состояния в ССК с принимающими состояниями определяет ***контрпример --*** вычисление, на котором **не** выполняется Φ.

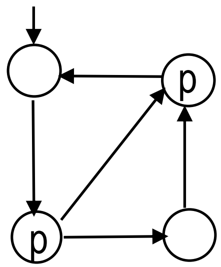




Пример: Model Checking для LTL

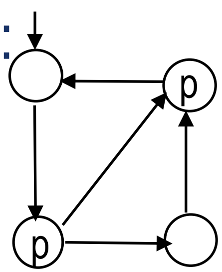
Удовлетворяет ли формуле Φ = GFp структура Крипке?

M::

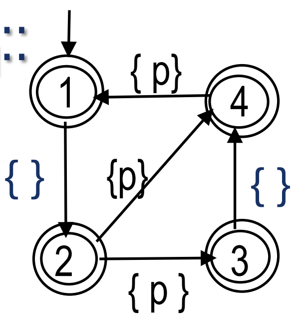


Если какое-либо вычисление *не удовлетворяет* Φ=GFp, то оно должно удовлетворять ¬Φ =¬GFp = FG ¬p

M::

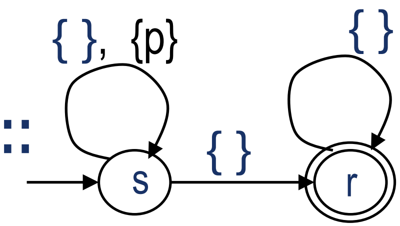


B_M::

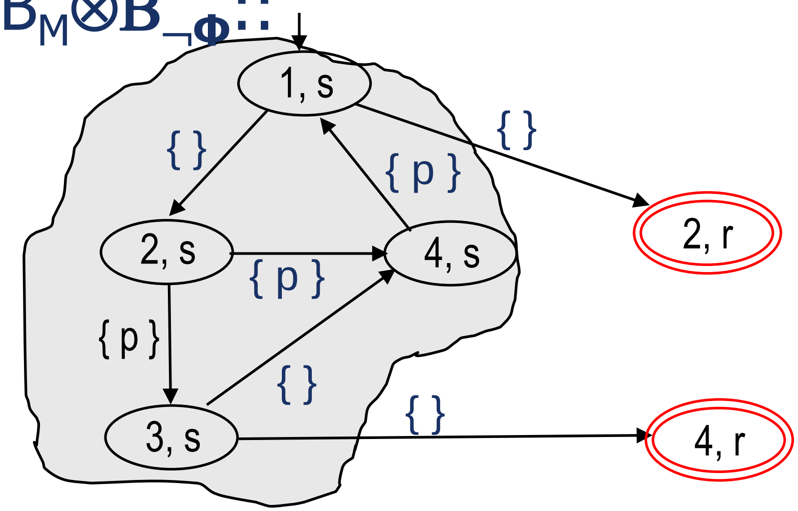


¬Φ = FG { }

$B_{\neg\Phi}$::



$B_M \otimes B_{\neg\Phi}$::



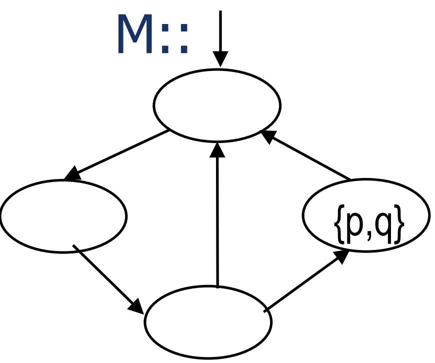
ССК без финальных состояний

Поскольку нет достижимой ССК, включающей **принимающее состояние**, язык $L_{BM \otimes B\neg\Phi}$ пуст. Следовательно, на структуре Крипке формула GFp **выполняется**

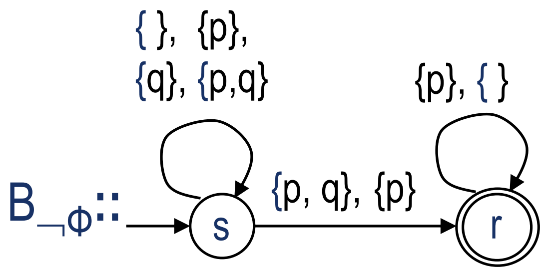
Model Checking для LTL: Пример

Выполняется ли формула Φ=G(p ⇒XFq) на структуре Крипке M?

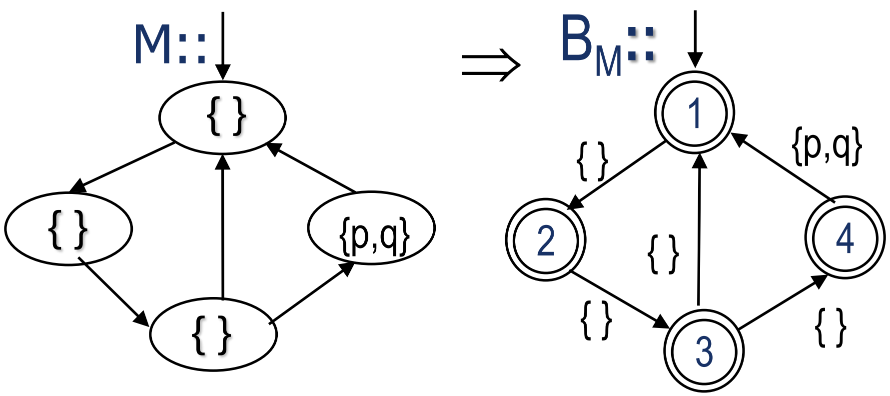
$\neg\Phi = \neg G(p \Rightarrow XFq) = F(\, p \wedge XG \neg q)$



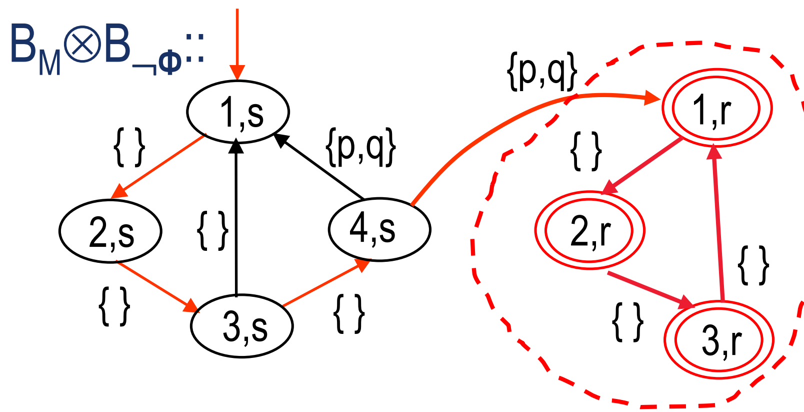
Автомат Бюхи $B_{\neg\Phi}$



Структура Крипке M ⇒ автомат Бюхи B_M



Композиция автоматов Бюхи:



Есть цикл, включающий принимающее состояние ⇒ язык $L_{BM \otimes B\neg\Phi}$ непуст.
M не удовлетворяет Φ.
Контрпример: $1234(123)^{\omega}$

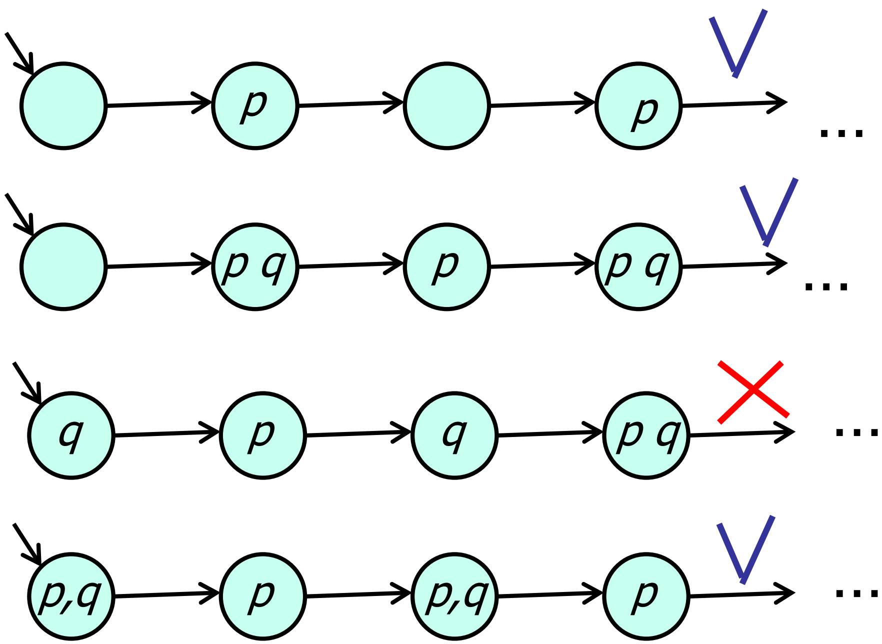
Некоторые факты об автоматах Бюхи и LTL

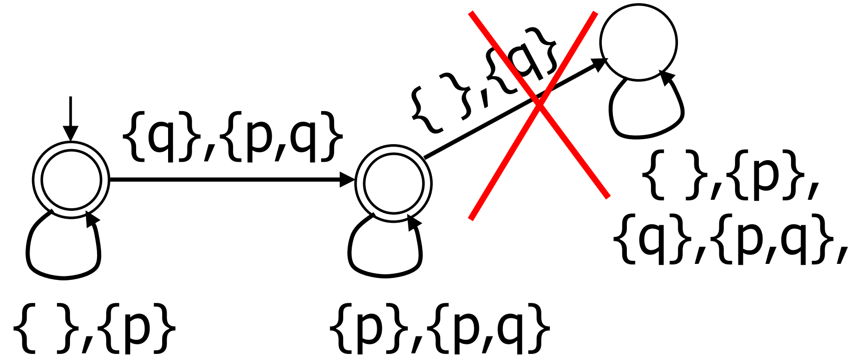
- Вопрос: почему сначала строят отрицание формулы LTL, а потом автомат Бюхи по отрицанию формулы?
  - Th. Язык автоматов Бюхи замкнут относительно операции дополнения.
    Иначе по любому автомату Бюхи можно построить автомат Бюхи, допускающий дополнение языка исходного автомата. Сложность получаемого автомата **экспоненциальна от размера исходного автомата**.
- Th. Язык автоматов Бюхи замкнут относительно операции асинхронной композиции
  - Доказательство очевидно
- Вопрос: а можно ли было использовать в качестве математической модели языка LTL структуру Крипке вместо автомата Бюхи?
  - Ответ очевиден, объяснение тоже

Обычно работаем с неполным автоматом Бюхи для формулы LTL

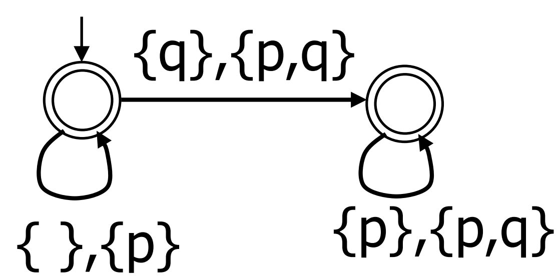
Автомат Бюхи, допускающий цепочки, определяемые формулой LTL G(q⇒XGp)

G(q⇒XGp)





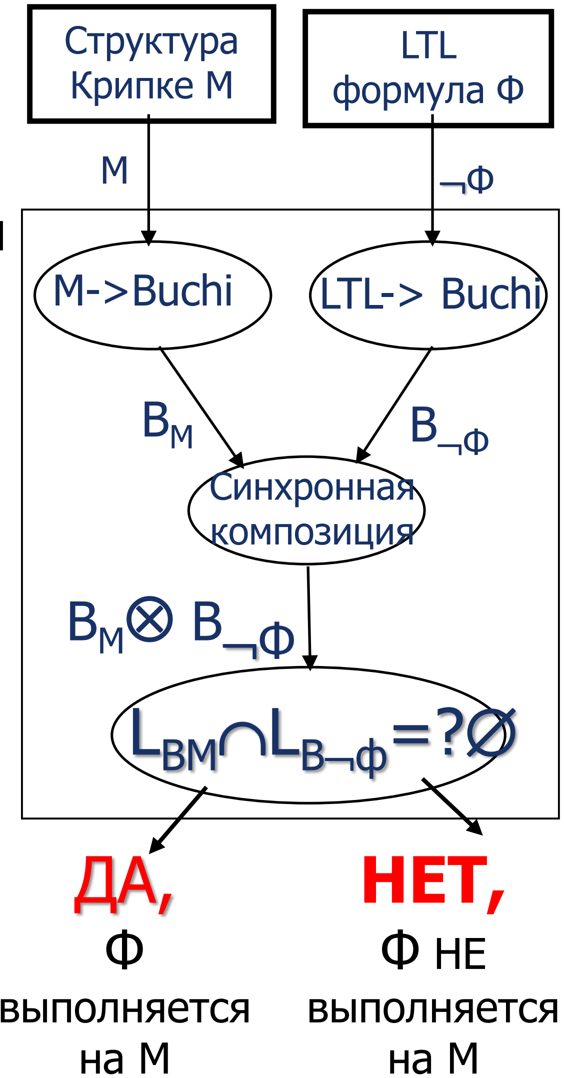
переходы в нефинальные состояния не указываем



Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

- 1. Построение по структуре Крипке M такого автомата Бюхи B_M, который допускает все возможные вычисления структуры M.
- 2. **Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Φ автомата Бюхи $B_Φ$**
- 3. Синхронная композиция двух автоматов Бюхи.
- 4. Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи.

  - Рассмотрим алгоритм построения автоматов Бюхи по LTL формуле, дающие более компактные автоматы



Автомат Бюхи: конечная модель ω-языков

- Автомат Бюхи B $=(Q, \Sigma, I, \delta, F)$

  Q – конечное множество состояний

  Σ - конечный алфавит

  $I \subseteq Q$ – множество начальных состояний

  $\delta \subseteq Q \times \Sigma \times Q$ – отношение переходов

  $F \subseteq Q$ – множество допускающих состояний

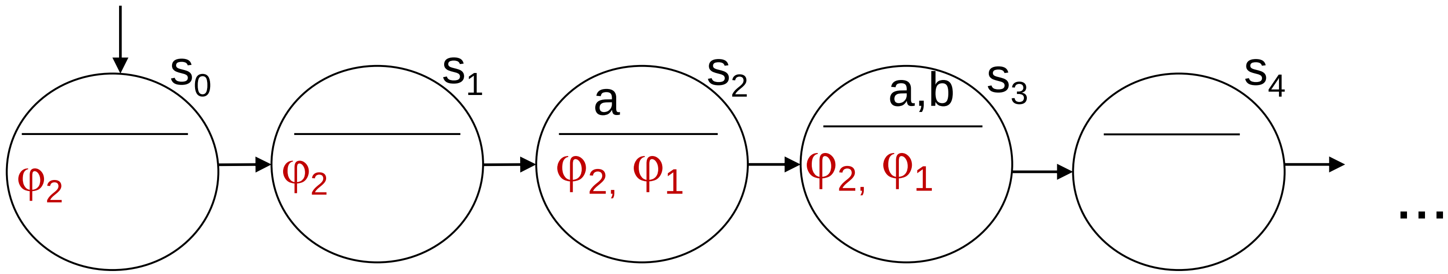
- Вычисление ρ автомата Бюхи B над ω-словом **w**$=a_0a_1\ldots \in \Sigma^\omega$ - бесконечная последовательность $\rho=q_0q_1 \ldots$ такая что:
  - $q_0 \in I$
  - $(\forall i \in N)\ (q_i\, a_i\, q_{i+1}) \in \delta$
- ρ является **допускающим** ттт $(\exists q \in F)\ q_i=q$ для бесконечного числа $i \in N$, $inf(\rho) \cap F \neq \varnothing$

- *Язык, индуцируемый автоматом Бюхи* $L_B \subseteq \Sigma^\omega$ - множество ω-слов, для которых существуют допускающие вычисления ρ автомата B

Разметка вычислений подформулами LTL

- LTL формула φ = **F**(a **U** b)
- Темпоральные подформулы: φ_1 = a **U** b, φ_2 = **F**(a **U** b)

Пример: рассмотрим вычисление $\rho=s_0s_1$ ... над словом w={}{}{a}{a,b}{}...



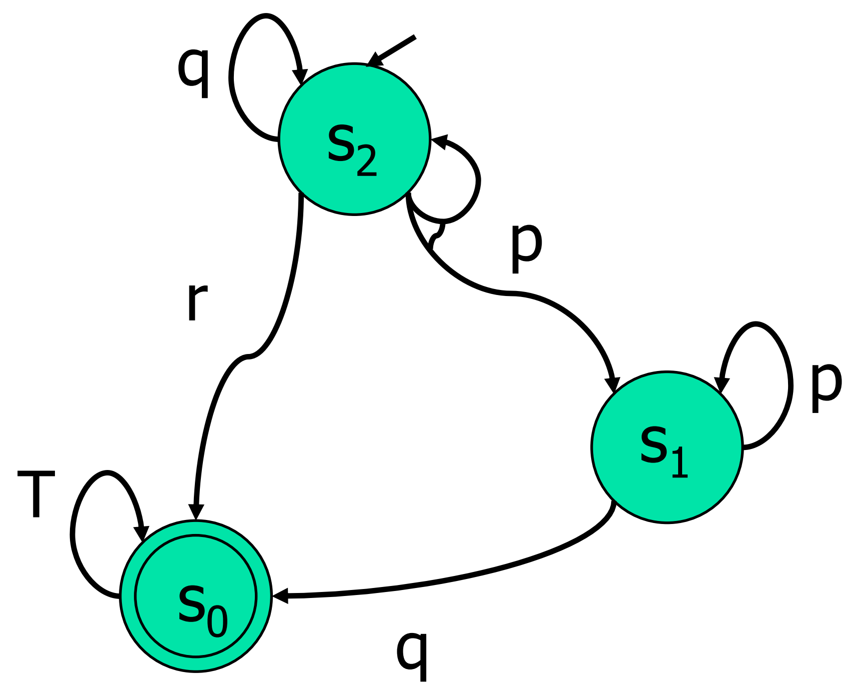
Состояние s помечается множеством темпоральных подформул φ, которые выполняются на вычислении, начинающимся в этом состоянии s

- *Th*. Для любой LTL формулы φ существует автомат Бюхи B, такой что $L_\varphi=L_B$
  - Размер автомата предыдущего алгоритма экспоненциален от размера формулы φ

LTL2BA с помощью альтернирующих автоматов

- Существуют более эффективные алгоритмы построения автомата Бюхи по LTL формуле, но они сложнее
- Один из таких алгоритмов на основе **альтернирующих автоматов**

Альтернирующий автомат содержит **И-** и **ИЛИ-переходы**

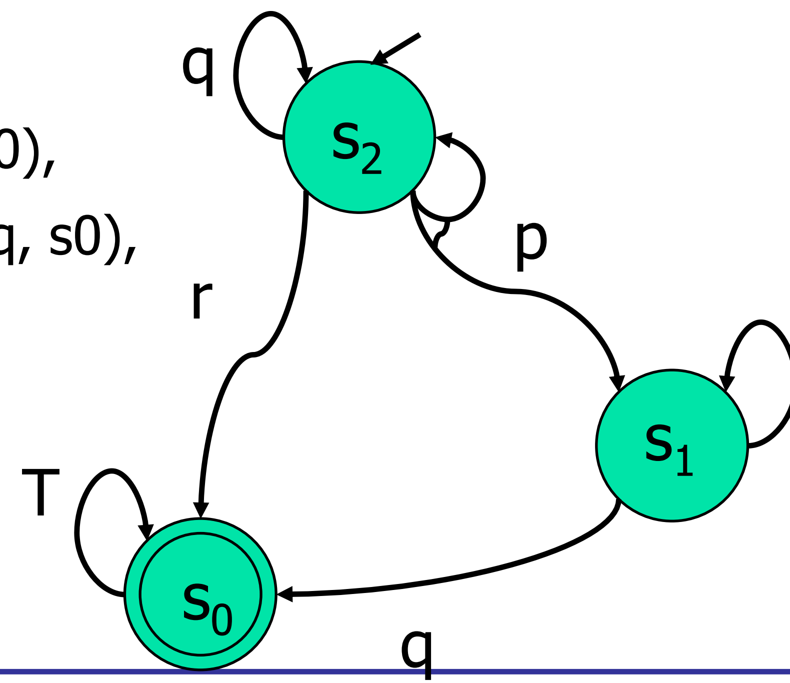


- По $\neg p \wedge q \wedge r$ из **s2** можно перейти в **s2** *ИЛИ* в **s0** (подобно обычным переходам конечных автоматов)
- По $p \wedge \neg q \wedge \neg r$ из **s2** автомат переходит в **s2** *И* в **s1**
- И-переходы обозначаются дугой

Альтернирующий автомат на бесконечных словах

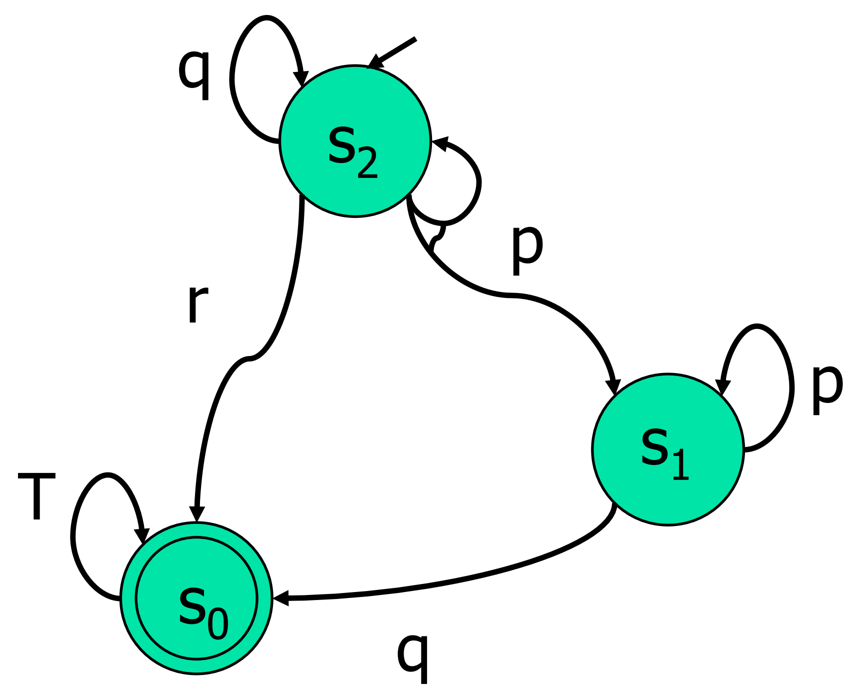
- Альтернирующий автомат $AA=(Q, \Sigma, I, \delta, F)$
  - Q – конечное множество состояний
  - Σ - конечный алфавит
  - $I \subseteq Q$ – множество начальных состояний
  - $\delta \subseteq Q \times \Sigma \times 2^Q$ – отношение переходов
  - $F \subseteq Q$ – множество допускающих состояний
- Отношение переходов:

  δ={(s2, q, s2), (s2, q, s0), (s2, r, s0), **(s2, p, (s1,s2))**, (s1, p, s1), (s1, q, s0), (s0, T, s0)}
- Допускающее состояние:
  - F={s0}



Табличная форма записи отношения переходов АА

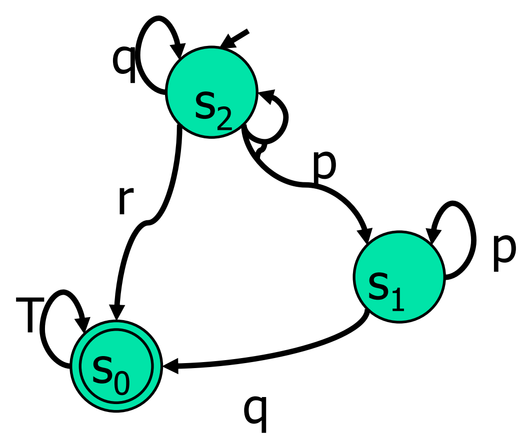
	{p}, {p,q},{p,r},{p,q,r}	{q},{p,q},{q,r}, {p,q,r}	{r}, {p,r},{q,r},{p,q,r}	{}
s2	(s2,s1)	s2	s0	
s1	s1	s1		
s0	s0	s0	s0	s0



(Почти) символьная форма записи отношения переходов АА

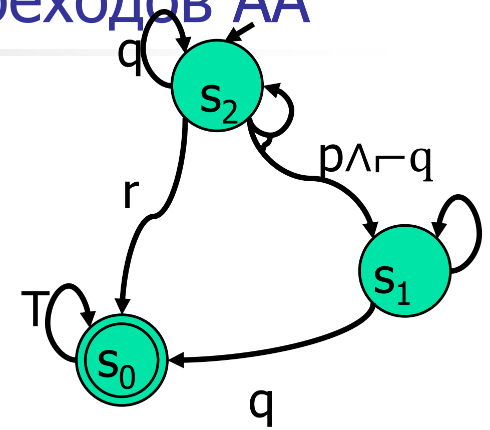
- Пометки на переходах – считаем за формулы
- Для каждого состояния вводим символ si
- ИЛИ-переходы обозначаем через дизъюнкцию
- И-переходы – через конъюнкцию
- Th. Состояния переходов АА образуют решетку без дополнения

p	q	r	δ(s2)	δ(s1)	δ(s0)
0	0	0	0	0	s0
0	0	1	s0	0	s0
0	1	0	s2	s0	s0
0	1	1	s2 \/ s0	s0	s0
1	0	0	s2/\s1	s1	s0
1	0	1	s2/\s1 \/ s0	s1	s0
1	1	0	s2/\s1 \/ s2	s1 \/ s0	s0
1	1	1	s2/\s1 \/ s2 \/ s0	s1 \/ s0	s0



Допустимое упрощение отношения переходов АА

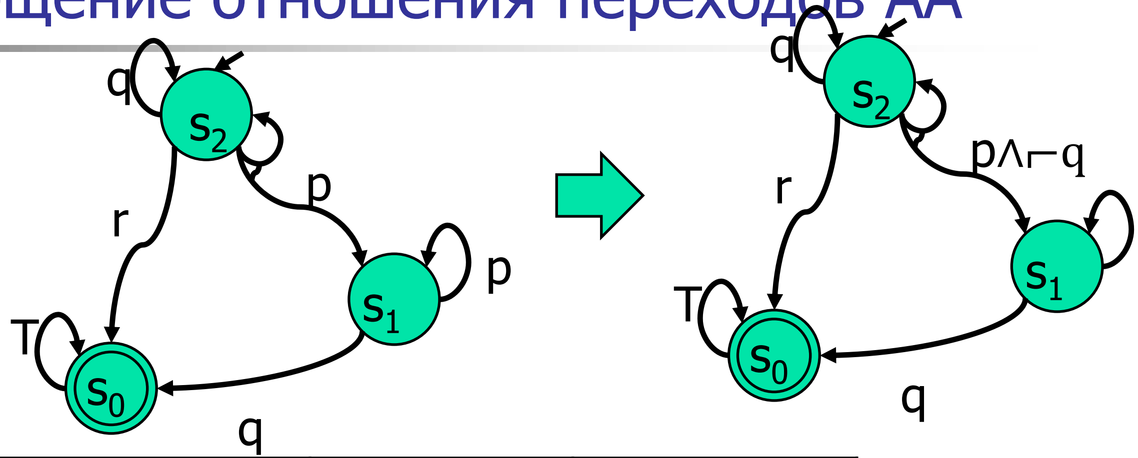
- Пользуемся правилом поглощения: a/\b \/ a = a



p	q	r	δ(s2)	δ(s2)	δ(s1)	δ(s0)
0	0	0	0	0	0	s0
0	0	1	s0	s0	0	s0
0	1	0	s2	s2	s0	s0
0	1	1	s2 \/ s0	s2 \/ s0	s0	s0
1	0	0	s2/\s1	s2/\s1	s1	s0
1	0	1	s2/\s1 \/ s0	s2/\s1 \/ s0	s1	s0
1	1	0	s2/\s1 \/ s2	s2	s1 \/ s0	s0
1	1	1	s2/\s1 \/ s2 \/ s0	s2 \/ s0	s1 \/ s0	s0

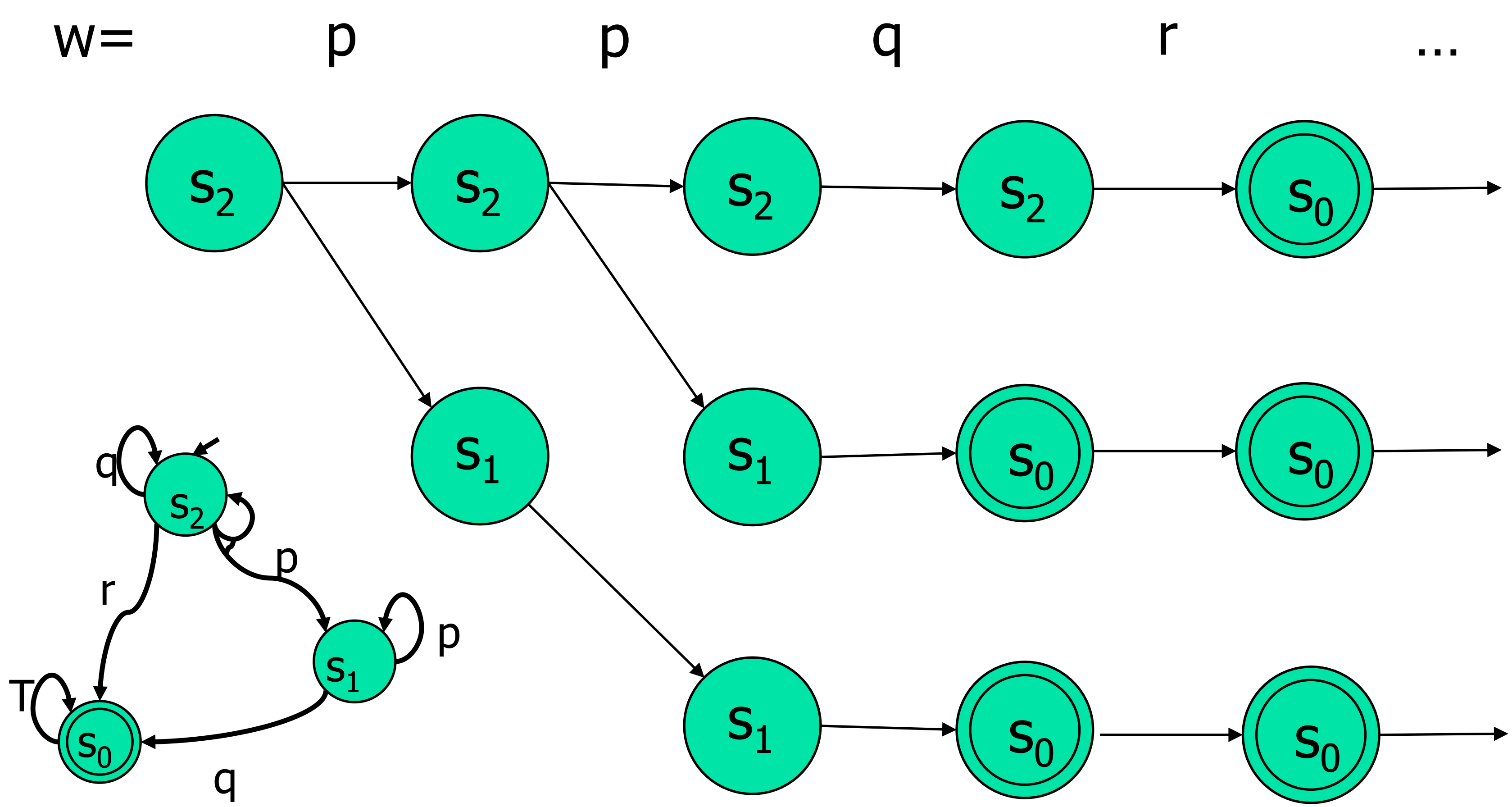
Допустимое упрощение отношения переходов АА

- Пометки на переходах увеличат детерминизм автомата
- Допускаемый язык не изменится



p	q	r	δ(s2)	δ(s2)	δ(s1)	δ(s0)
0	0	0	0	0	0	s0
0	0	1	s0	s0	0	s0
0	1	0	s2	s2	s0	s0
0	1	1	s2 ⋁ s0	s2 ⋁ s0	s0	s0
1	0	0	s2/\s1	s2/\s1	s1	s0
1	0	1	s2/\s1 ⋁ s0	s2/\s1 ⋁ s0	s1	s0
1	1	0	s2/\s1 ⋁ s2	s2	s1 ⋁ s0	s0
1	1	1	s2/\s1 ⋁ s2 ⋁ s0	s2 ⋁ s0	s1 ⋁ s0	s0

Вычисление альтернирующего автомата – в общем случае - дерево



Вычисление альтернирующего автомата на бесконечном слове w

$\rho = \{V, D\}$ – **вычисление** AA $=(Q, \Sigma, I, \delta, F)$ – бесконечное дерево

- $V = \cup V_i$, бесконечное множество вершин, расположенных по уровням, V_i – множество вершин на i-м уровне
- D – отношение переходов:
- $s = V_0 \wedge s \in I$
  - корень дерева единственный, одна из начальных вершин AA
- $\forall s \in V_i, \exists q \in V_{i+1}. (s, w_i, q) \in D$
  - для каждой вершины существует последователь на следующем уровне
- $\forall q \in V_{i+1}, \exists_1\, s \in V_i. (s, w_i, q) \in D, i \geq 0$
  - для каждой вершины существует родитель на предыдущем уровне и только один, кроме корня
- $\forall s \in V_i, \forall q \in V_{i+1}. \left((s, w_i, q) \in D \rightarrow \exists C \subseteq Q. \left((s, w_i, C) \in \delta \wedge q \in C\right)\right)$
  - Если есть переход в вычислении, то существует переход, который соответствует ему в AA
- $\forall s \in V_i, \forall q \in V_{i+1}. \left((s, w_i, q) \in D \& \exists C \subseteq Q. \left((s, w_i, C) \in \delta \wedge q \in C\right) \rightarrow \forall r \in C. r \in V_{i+1} \wedge (s, w_i, r) \in D\right)$
  - Переход в вычислении должен содержать переход во все вершины, соответствующие переходу AA
- $\forall s, s' \in V_i, \forall q, q' \in V_{i+1}, \forall w_i, w_i'. \left((s, w_i, q) \in D \wedge (s', w'_i, q') \in D \rightarrow w_i = w'_i\right)$
  - Переход на одном уровне осуществляется по одинаковым символам

Допускающее условие по Бюхи

Ветвь β вычисления ρ автомата АА на бесконечном слове w – последовательность состояний, $\beta = \beta_0\ \beta_1\ \beta_2\ \beta_3 \ldots$

- где каждое β_i является состоянием вычисления ρ
- каждый переход (β_i, β_{i+1}) – переход вычисления ρ

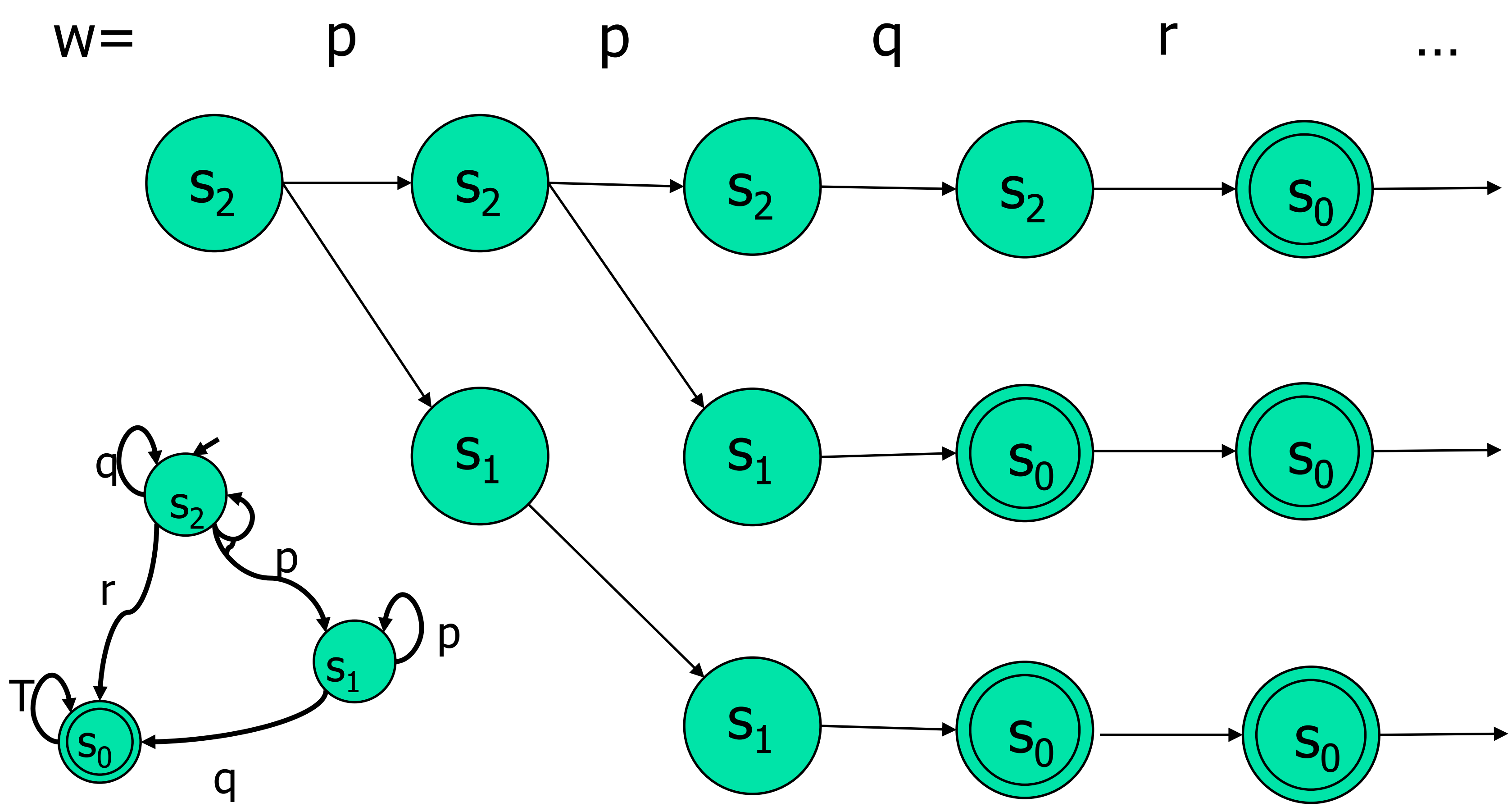
Ветвь β вычисления ρ автомата АА называется **допускающей по Бюхи**, ттт когда на ней бесконечное количество раз встречается допускающее состояние АА, $inf(\beta) \cap F \neq 0$

Вычисление ρ автомата АА на бесконечном слове w называется **допускающим по Бюхи** ттт, когда все его ветви являются допускающими

Слово w **допускается АА** ттт, когда существует допускающее вычисление ρ

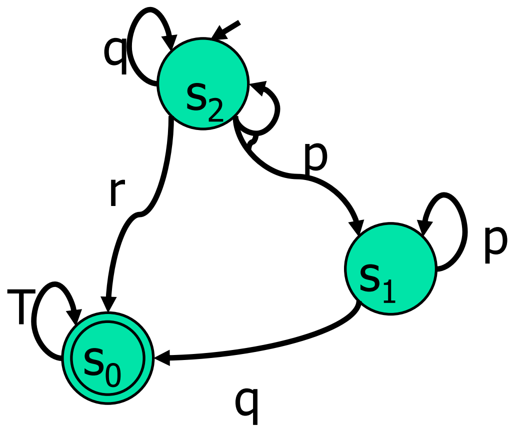
Множество всех бесконечных слов, допускаемых альтернирующим автоматом АА, называется **языком АА**

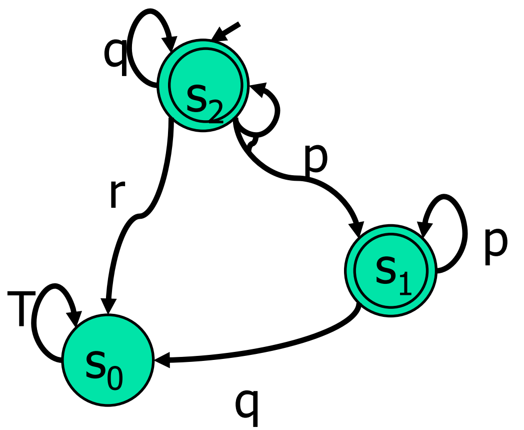
Допускающее вычисление альтернирующего автомата по Бюхи



Свойства альтернирующего автомата

- Th. По любой LTL можно построить АА такой, что он допускает язык, индуцируемый LTL-формулой и только его
- Th. Языки АА замкнут относительно операции дополнения. Сложность построения альтернирующего автомата, допускающего дополнение языка АА, линейна относительно размера автомата
  - Для автомата Бюхи сложность построения экспоненциальна!





Перевод в отрицательную нормальную форму

- Грамматика LTL в отрицательной нормальной форме (ОНФ)

  φ :: = p | ¬p | φ∨φ| φ∧φ | φ**U**φ | φ**R**φ | **X**φ

- $a\ R\ b$ – а отпускает b, R – Release, $a\ R\ b = \neg(\neg a\ U \neg b)$
- Отрицательная нормальная форма – это когда отрицания стоят только перед атомарными высказываниями
- Th. Любую LTL формулу можно перевести в отрицательную нормальную форму
- Свойства, позволяющие перевести в ОНФ

$$\neg(a \vee b) = \neg a\ \wedge \neg b$$
$$\neg(a \wedge b) = \neg a \vee \neg b$$
$$\neg(aUb) = \neg a\ R \neg b$$
$$\neg(aRb) = \neg a\ U \neg b$$

- Пример: $Gp = \neg F(\neg p) = \neg(T\ U \neg p) = \bot\ R\ \neg\neg p = \bot\ R\ p$

Алгоритм построения альтернирующего автомата по LTL формуле

- Состояниями будут:
  - состояния, помеченные темпоральными подформулами формулы LTL φ, т.е. X, U, R
  - + 1 состояние, помеченное T=true, переходы из этого состояния по любому символу приводят в него самого
  - + 1 состояние, помеченное самой формулой φ, если в корне синтаксического дерева φ стоит нетемпоральная формула
- Начальным состоянием будет состояние, помеченное φ
- Допустимыми по Бюхи будут состояния, помеченные T, R
  - Если в АА нет И-переходов, то он будет совпадать с автоматом Бюхи. В этом случае имеет смысл рассматривать условия по Бюхи
- Допустимыми по ко-Бюхи будут состояния, помеченные U
  - Если в АА есть И-переходы, то имеет смысл рассматривать условия по ко-Бюхи

Алгоритм построения альтернирующего автомата по LTL формуле

Введем булевы символы для обозначения состояний:

- $s_{\varphi\, U\, \psi}$- состояние, соответствующее формуле $\varphi\; U\; \psi$
- $s_{\varphi\, R\, \psi}$ - состояние, соответствующее формуле $\varphi\; R\; \psi$
- s_{φ}- состояние, соответствующее формуле $\mathrm{X}\varphi$

- Построение переходов основано на свойствах распространения обязательств
  - $\varphi\; U\; \psi = \psi \vee \varphi \wedge X(\varphi\; U\; \psi)$
  - $\varphi\; R\; \psi = \psi \wedge \big(\varphi \vee X(\varphi\; R\; \psi)\big)$

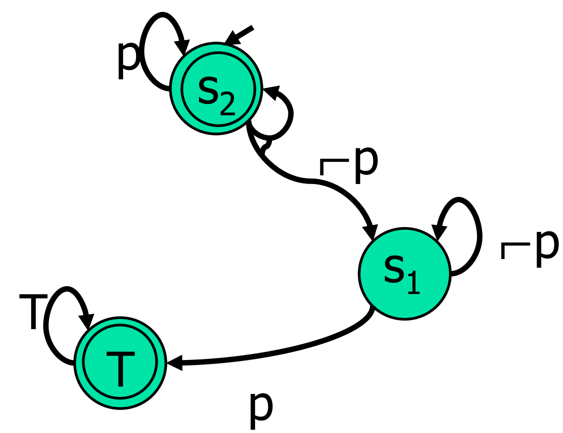
Переходы получаются для каждой формулы с помощью двоичных функций

- $\delta(p) = p$
- $\delta(\neg p) = \neg p$
- $\delta(X\varphi) = s_{\varphi}$
- $\delta(\varphi\; U\; \psi) = \delta(\psi) \vee \delta(\varphi) \wedge s_{\varphi\, U\, \psi}$
- $\delta(\varphi\; R\; \psi) = \delta(\psi) \wedge \big(\delta(\varphi) \vee s_{\varphi\, R\, \psi}\big)$

Примеры. Gp, Fp, aUb, aRb

- $\varphi = G\, F\, p$
- Приводим в отрицательную нормальную форму: $\varphi = \perp R\, (T\, U\, p)$
- Подформулы этой формулы: $f_0 = p, f_1 = T\, U f_0,\ f_2 = \perp R\ f_1$
- Состояния АА: s_1 состояние по формуле f_1, s_2 состояние по формуле f_2, T состояние по формуле T
- Начальное состояние s_2
- Допускающее состояние по Бюхи s_2, T
- Отношение переходов:

p	δ_0	δ_1	δ_2
0	0	s_1	$s_1 \wedge s_2$
1	1	1	s_2

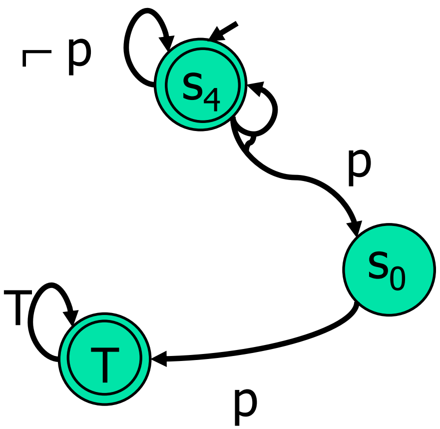


- Очевидно, что δ_0 можно не указывать, так как оно совпадает со столбцом p и здесь f0 не образует состояния

Пример 2

- $\varphi = G(p \to Xp)$
- Приводим в отрицательную нормальную форму: $\varphi = \perp R\, (\neg p \,\vee Xp)$
- Подформулы этой формулы: $f_0 = p$, $f_1 = Xf_0$, $f_2 = \neg p$, $f_3 = f_2 \vee f_1$, $f_4 = \perp R\, f_3$
- Состояния АА: s_0 состояние по формуле f_1, s_4 состояние по формуле f_4, T состояние по формуле T
- Начальное состояние s_4
- Допускающее состояние по Бюхи s_4, T
- Отношение переходов:

p	δ_0	δ_1	δ_2	δ_3	δ_4
0	0	s_0	1	1	s_4
1	1	s_0	0	s_0	$s_0 \wedge s_4$



- здесь f0 образует состояние!
- $\delta_1, \delta_2, \delta_3$ - необходимы только для построения перехода δ_4

Тонкие абстрактные модели

- темпоральная логика,

- ω -языки,

- недетерминированные автоматы Бюхи,

- синхронная композиция автоматов Бюхи, ...,

которые реализовать “в железе” вообще невозможно, позволили построить алгоритмы проверки свойств поведения реальных сложных технических систем: коммуникационных протоколов, драйверов, систем логического управления, бортовых систем космических аппаратов и т.п.

Заключение

- Некоторые свойства систем НЕ выражаются CTL-формулами, но выражаются LTL-формулами. Поэтому нужны и алгоритмы проверки выполнимости таких формул на структуре Крипке
- Для проверки того, является ли M моделью формулы Φ логики LTL, строятся автоматы Бюхи A_M и $B_{\neg\Phi}$, и проверяется пустота языка, допускаемого автоматом Бюхи - синхронной композицией $A_M \otimes B_{\neg\Phi}$
- Сложность алгоритма проверки моделей для LTL- формул значительно выше, чем для CTL формул: $O(\ |A| * 2^{|\Phi|}\ )$. Но формулы обычно малы!
- **Получение контрпримера в результате выполнения алгоритма model checking имеет огромное значение для отладки технических систем**
- Большинство инструментальных систем верификации выполняет алгоритм проверки модели для CTL.
- Система Spin конструирует $B_{\neg\Phi}$ и проверяет, выполняется ли заданная LTL формула на введенной модели.

Спасибо за внимание