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

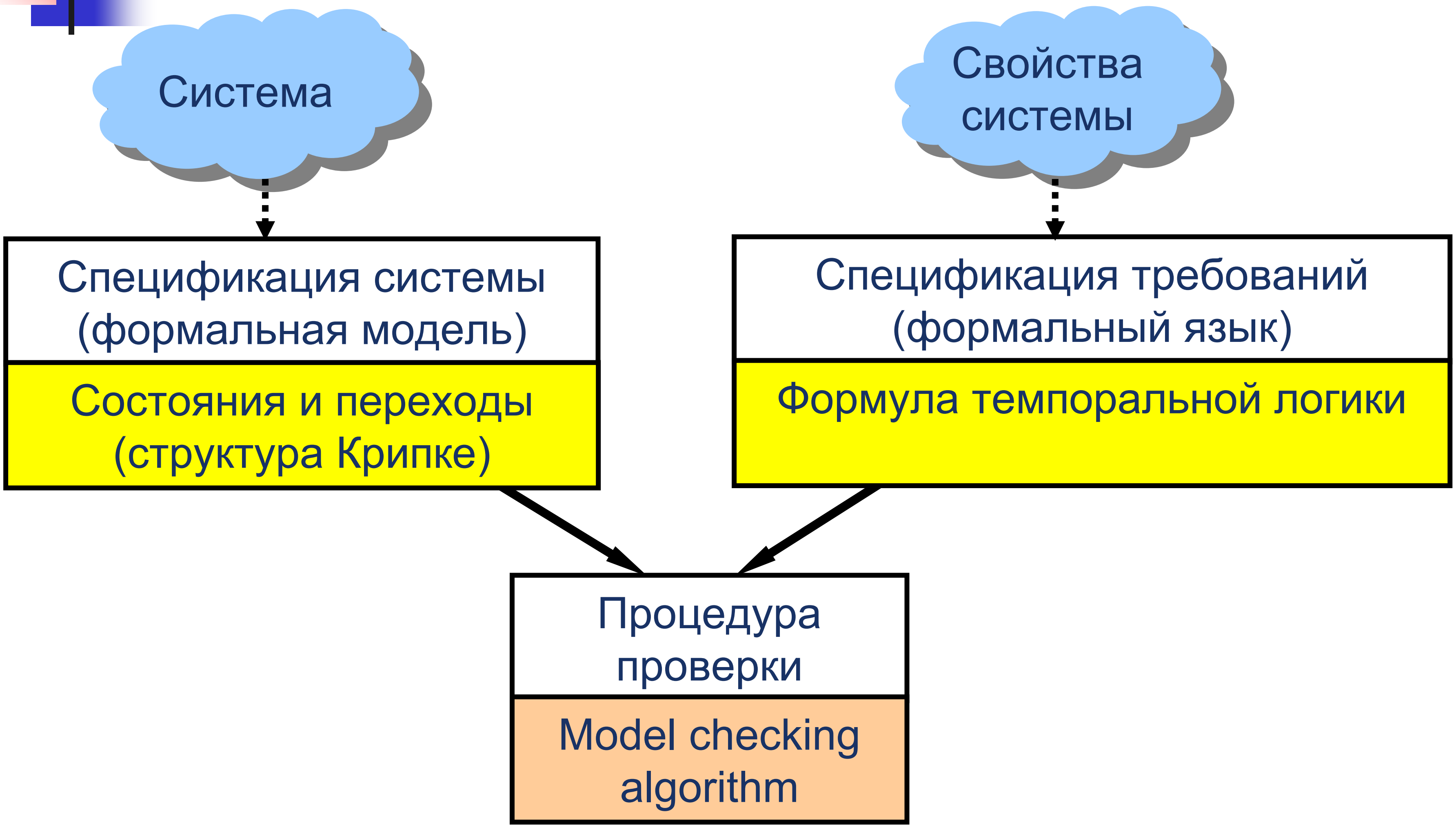
План курса

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 6

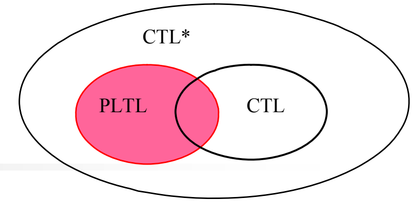
Model checking для формул LTL

Model Checking



Наша задача – рассмотреть алгоритм Model checking для логики LTL

Сравнение логик LTL, CTL* и CTL



Формулы LTL:

$A\ G[\neg p \ \& \ \neg Fq \Rightarrow r\ ]$

$A\ [\neg a \vee Gb \ \& \ (aU\ \neg c)]$

$A\ [a\ U\ (G \neg\ b)\ ]$

Формулы CTL:

$AG[\ p \& \neg EF(q \Rightarrow r)\ ]$

$AF[\ a \ \& \ E(aU\ \neg c)]$

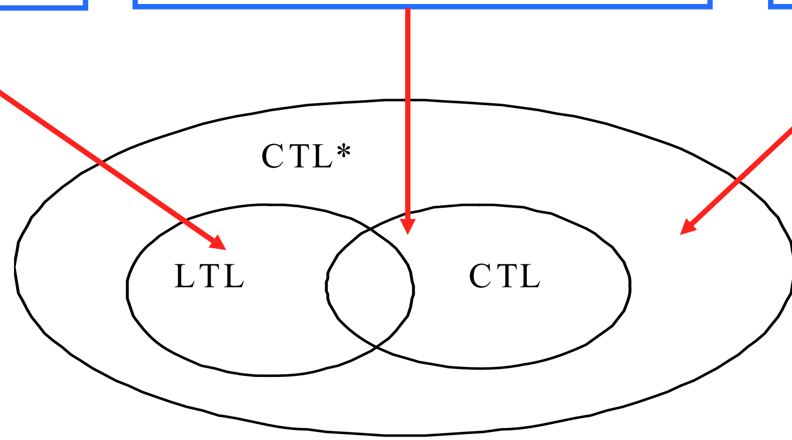
$E\ [a\ U\ (AG \neg\ b)\ ]$

Формулы CTL*:

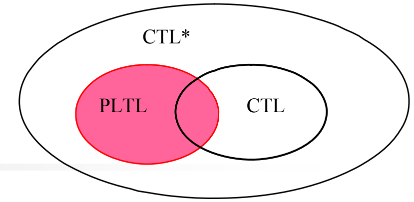
$EG[\neg p \ \& \ X\ A\ F(q \Rightarrow r\ )]$

$E\ F[\ a \ \& \ (\ aU\ \neg c)\ ]$

$AF[a\ U\ (AG \neg\ b)\ ]$



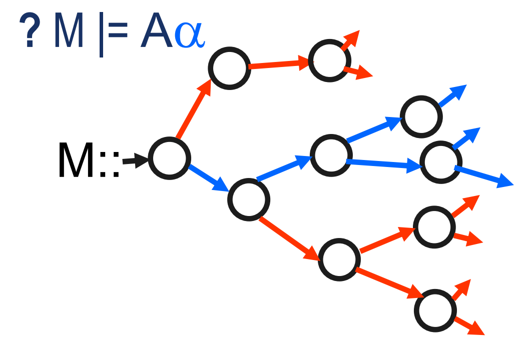
LTL – подмножество CTL*



Почему важна проверка LTL-формул? Потому что некоторые свойства систем НЕ выражаются CTL-формулами, но выражаются LTL-формулами

Формулы LTL строятся как Aα,
где α - формула пути

α ::= p |$\neg\alpha$ | $\alpha\vee\alpha$ | $\alpha U \alpha$ | Xα



Операторы G и F определяются через Until

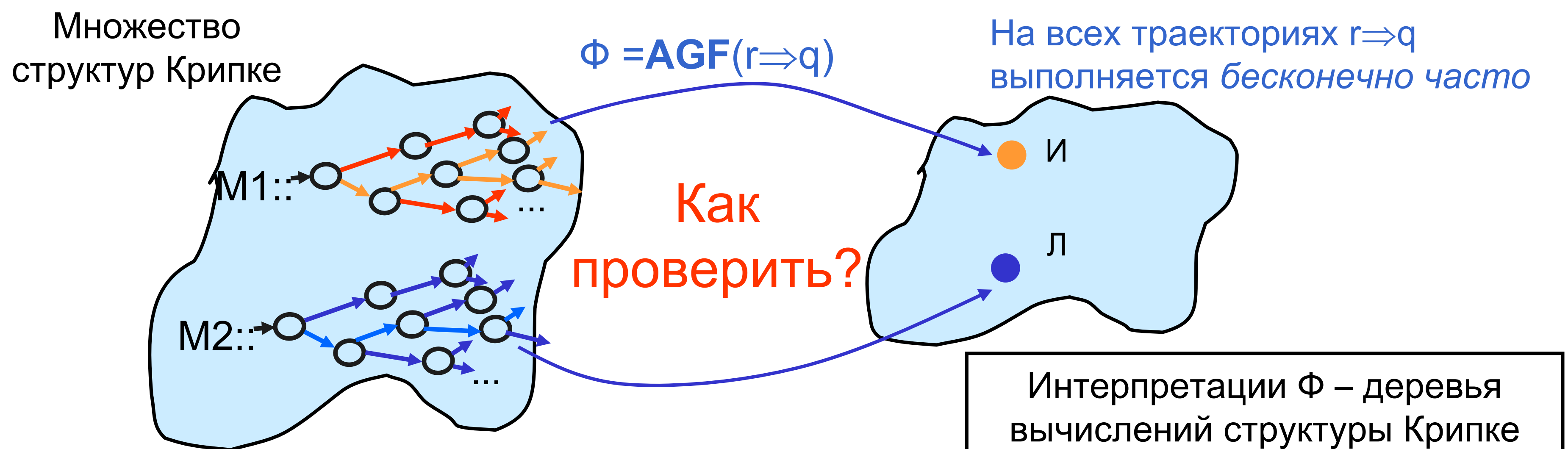
$F\alpha = True\ U\ \alpha \quad G\ \alpha = \neg F \neg \alpha$

Проверка LTL- формулы для M требует рассмотрения всех вычислений M, а таких вычислений бесконечное число, и каждое из них бесконечно

Формула Линейной Темпоральной Логики выполняется для структуры Крипке, если она выполняется ***для любого пути***, начинающегося в начальном состоянии структуры Крипке

Model checking для LTL

LTL – не state формулы (как в CTL), алгоритм маркировки использовать нельзя



Интерпретации Ф – деревья вычислений структуры Крипке

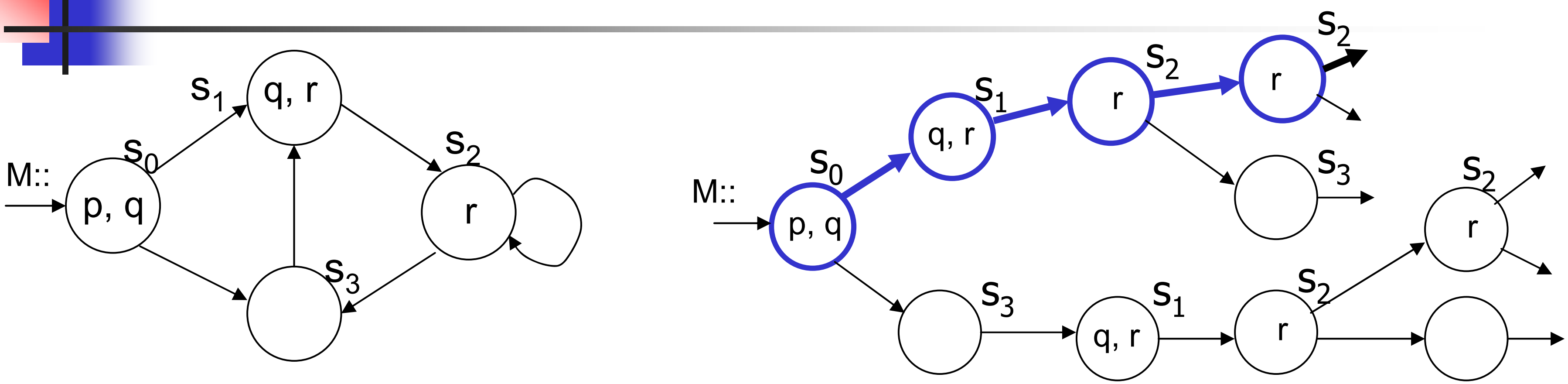
Чтобы формула LTL была истинной на структуре Крипке, нужно, чтобы она была истинной на всех вычислениях структуры Крипке

Формула LTL будет ложной на структуре Крипке M, если даже только одно вычисление M не удовлетворяет этой формуле

Перебирать и проверять все пути бессмысленно

Как построить алгоритм model checking для LTL?

Пример траекторий структуры Крипке



Свойства поведения формулируются не для вычислений (последовательностей состояний системы), а для траекторий (последовательностей подмножеств атомарных предикатов системы)

Возможные траектории M:

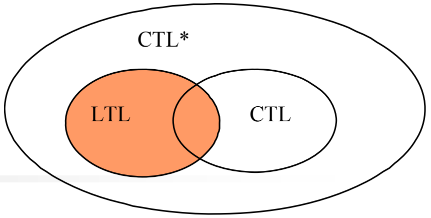
{p,q} {q,r} {r} {r} {r} {r} ... Пример свойства: $(p \vee q)U(Gr)$

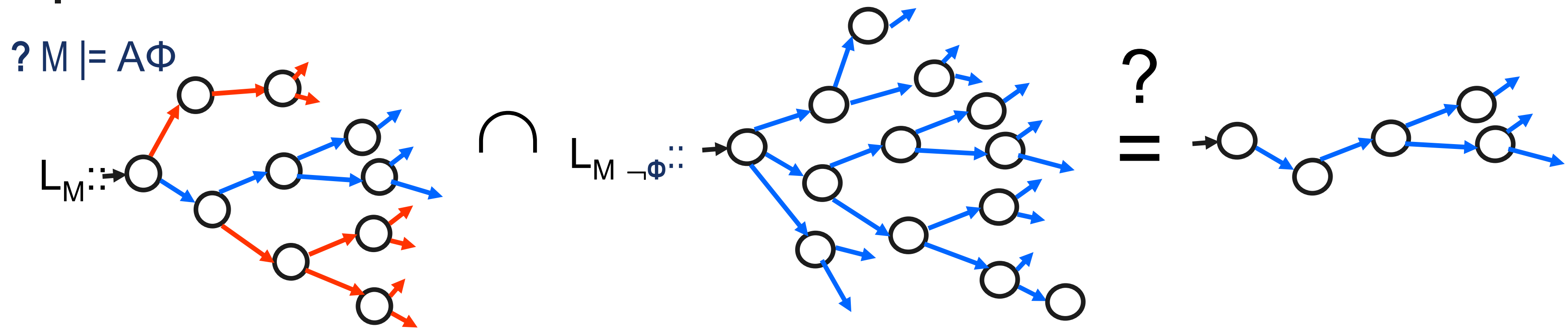
{p,q} { } {q,r} {r} { } {q,r} {r} ...

{p,q} {q,r} {r} { } { } { } ...

Траектории бесконечны, и самих траекторий бесконечное число

Model Checking для LTL





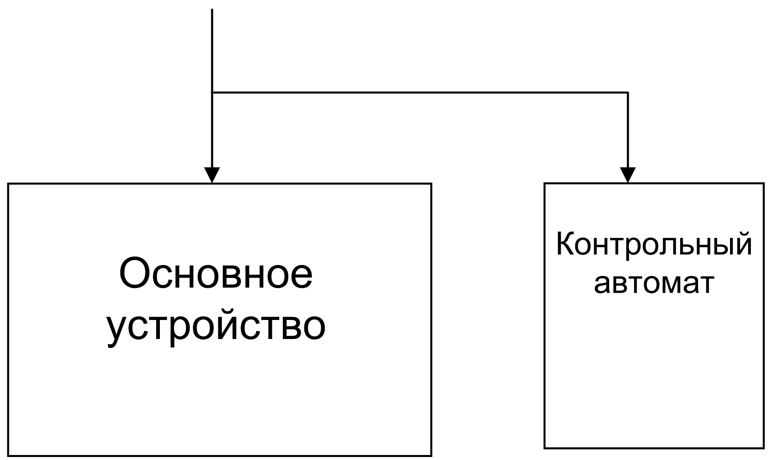
- Единственный контрпример достаточен, чтобы опровергнуть M |= AФ, где Ф – формула LTL, а квантор A говорит о том, что Ф выполняется на ВСЕХ траекториях структуры Крипке M
- Можно проверить, совпадают ли какие-нибудь вычисления в M с вычислениями, удовлетворяющими ¬Ф

Идея подхода к верификации

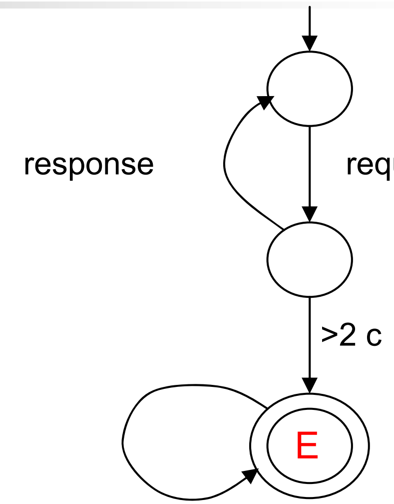
Конечным образом опишем все цепочки языка, которые НЕ удовлетворяют Ф, и проверим, есть ли пересечения с траекториями (языком) M.

Описание возможных траекторий удобно сделать с помощью автомата

Контрольный автомат



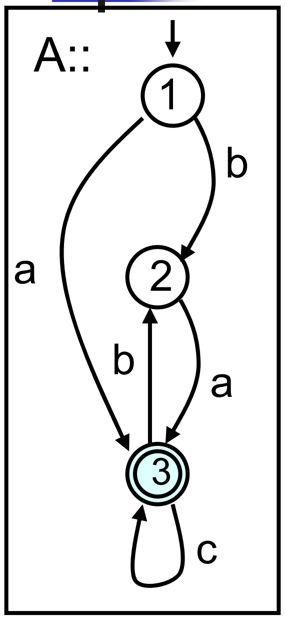
Контроль правильности функционирования системы с помощью контрольного автомата (watchdog)



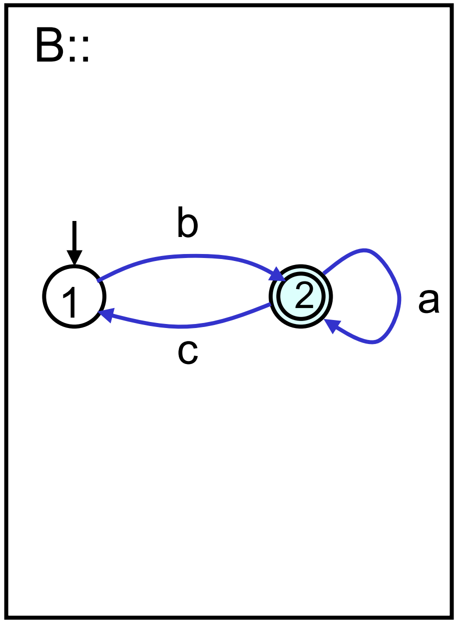
Переход в состояние E (Error), если ответ на любое посланное сообщение придет позже, чем через 2 с

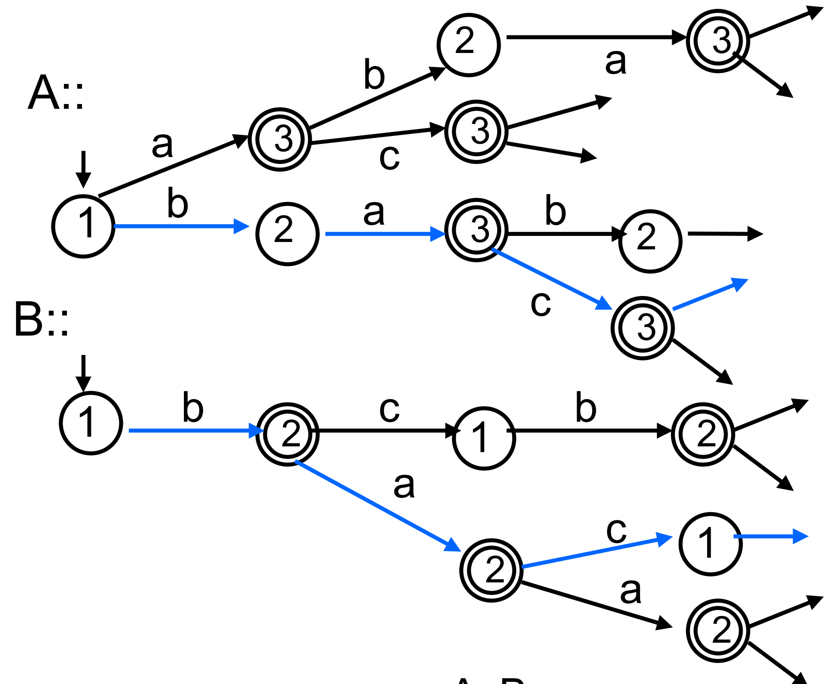
Подобная техника будем использовать для проверки выполнения LTL формул на структуре Крипке. Он называется “теоретико-автоматным подходом” - один из наиболее удобных подходов к верификации LTL

Как найти пересечение языков: Конечные Автоматы

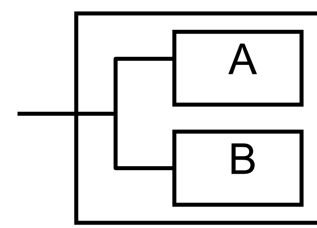


$\otimes$





AxB



L_A= {ba, acc, bacba, ...}

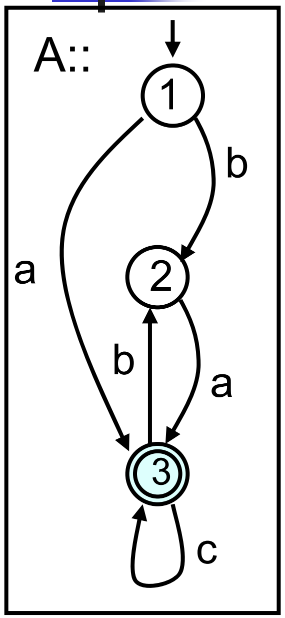
L_B ={b, ba, bacb, bacba, ...}

$L_A \cap L_B$ =? – непросто!

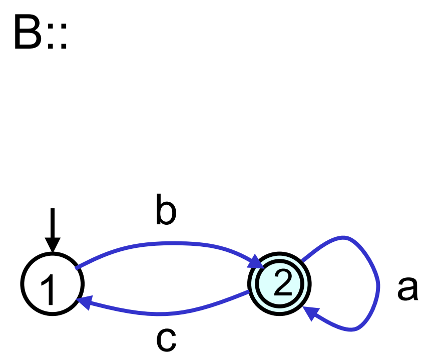
$L_A \cap L_B = L_{A \otimes B}$

Язык, являющийся пересечением автоматных языков, можно определить, построив синхронную композицию конечных автоматов

Как найти пересечение языков: КА

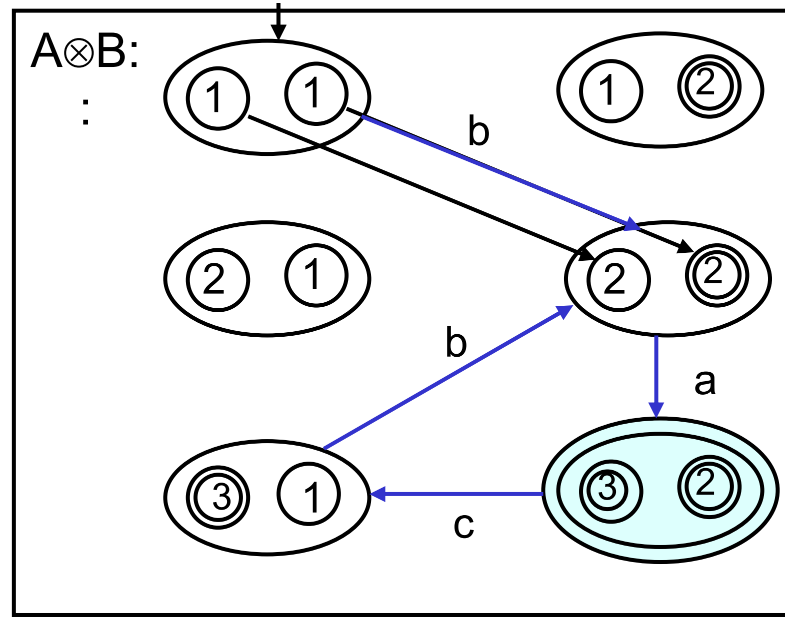


⊗



Пусть B – автомат, задающий все ошибочные цепочки

=

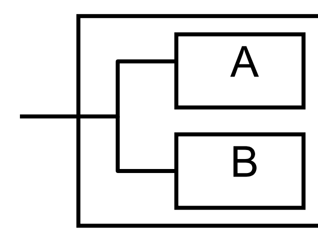


L_A= {ba, acc, bacba, ...}

L_B ={b, ba, bacb, bacba, ...}

$$L_A \cap L_B = L_{A \otimes B}$$

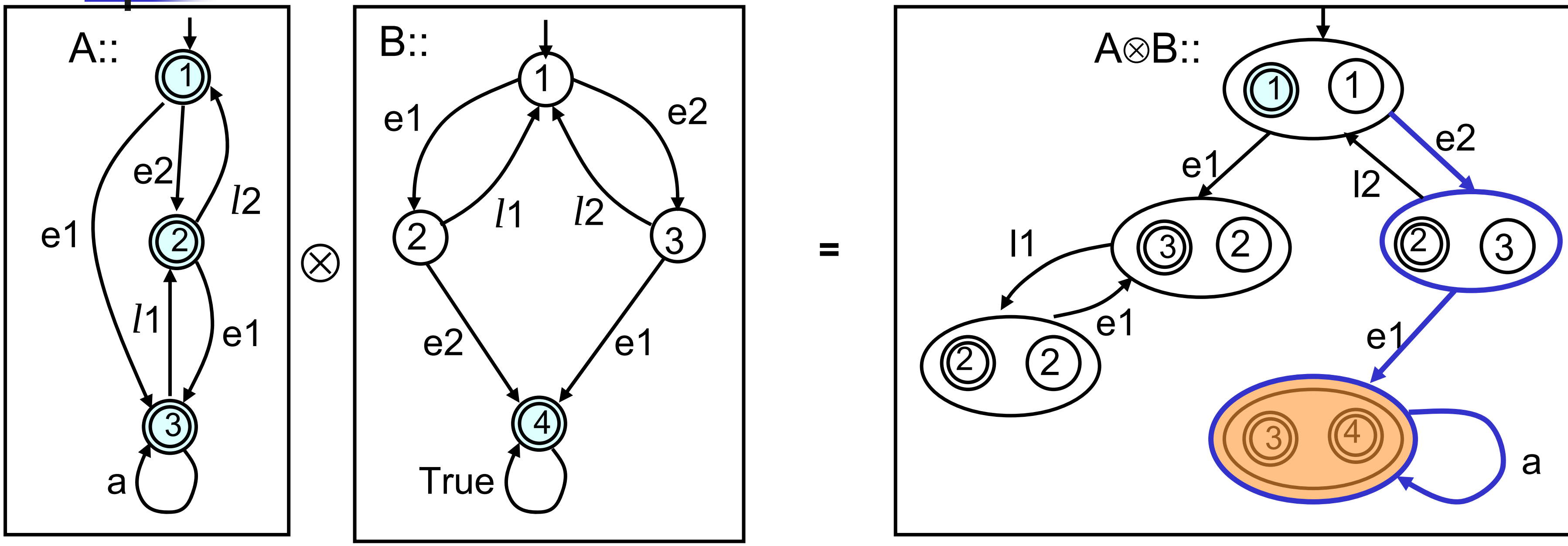
A⊗B



- Синхронная композиция автоматов – это два автомата, стоящие рядом
- Язык, допускаемый A⊗B – это пересечение языков, допускаемых A и B

Пример: взаимное исключение

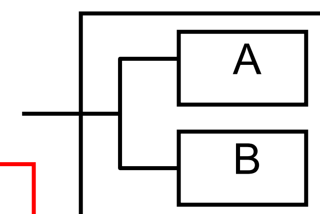
Синхронное произведение



ei – вход процесса i в крит интервал
*l*i – выход процесса i из крит интервала

$L_A \cap L_B \neq \varnothing$

A⊗B



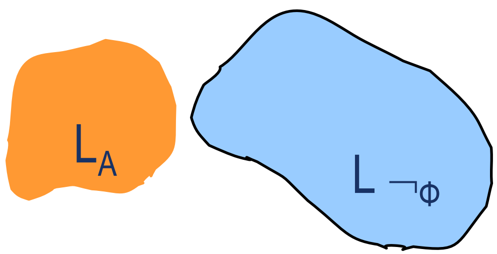
A - автомат, описывающий поведение проектируемой системы процессов

B – контрольный автомат, задающий все ошибочные цепочки: вход в критический интервал двух процессов

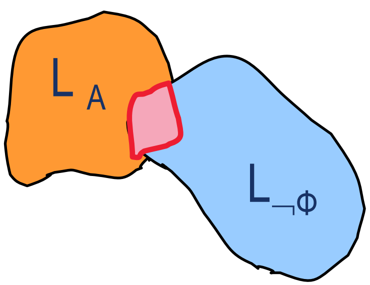
Нашли некорректность на стадии проектирования!

Model Checking для LTL – анализ множеств траекторий

- Пусть L_A - множество всех траекторий (последовательностей пометок в состояниях вычислений) системы A. L_A назовем языком системы A
- Пусть $L_{\neg\Phi}$ – множество всех НЕправильных траекторий, т.е. тех, которые не соответствуют проверяемому свойству Φ
- Тогда A корректна (относительно свойства Φ), если $L_{\neg\Phi} \cap L_A = \varnothing$ (т.е. каждая цепочка языка A принадлежит множеству правильных траекторий), или, что то же, если язык $L_{\neg\Phi}$ не пересекается с L_A:



В структуре Крипке A свойство Φ выполняется



В структуре Крипке A свойство Φ НЕ выполняется

Но языки L_A и $L_{\neg\Phi}$ нужно как-то описать конечным образом

Автоматы Бюхи

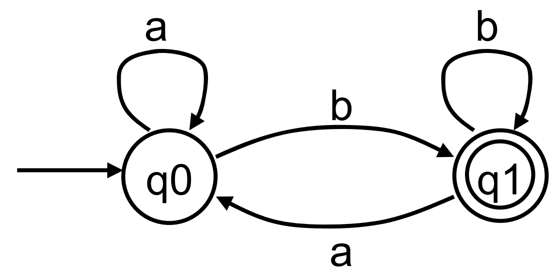
Как представить конечным образом множество бесконечных цепочек?

LTL-формула конечным образом описывает множество бесконечных траекторий, ей удовлетворяющих. Конечные автоматы также конечным образом задают множество траекторий (последовательностей цепочек). Но конечные автоматы-распознаватели языков работают с конечными цепочками. Нужен новый тип автомата, распознающего БЕСКОНЕЧНЫЕ последовательности.

Такой автомат был введен Richard Buchi

Автомат Бюхи – автомат, распознающий бесконечные цепочки. Цепочка допускается автоматом Бюхи B, iff существует заключительное состояние автомата, которое проходится бесконечное число раз при приеме этой цепочки

Теория формальных языков построена для анализа КОНЕЧНЫХ цепочек



Конечный Автомат: допускается любая цепочка, оканчивающаяся на b:

(a+b)*b

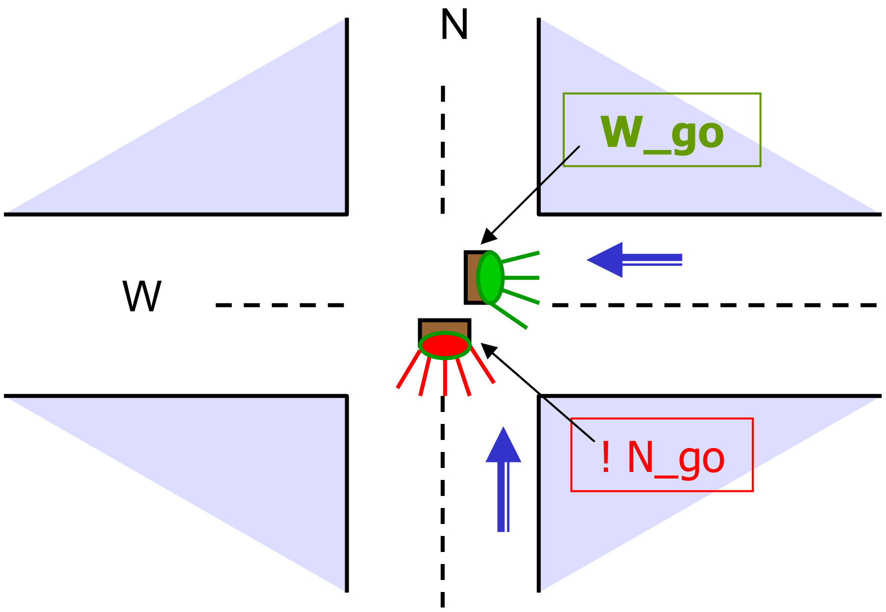
Автомат Бюхи: допускаются цепочки, в которых b встречается бесконечно много раз :

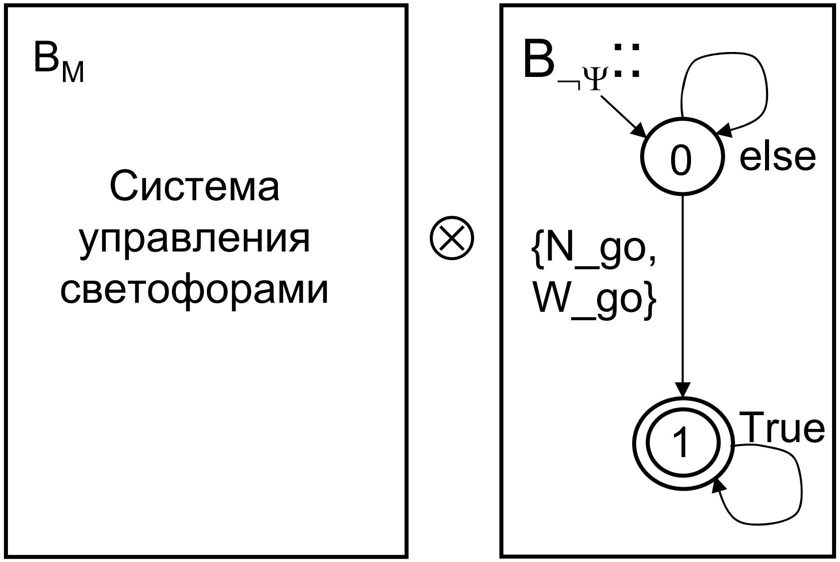
GFb

Пример контрольного автомата

Требование к системе управления светофорами: не разрешать движения по пересекающимся путям, например, в северном и западном: $\Psi = \neg\mathbf{F}(N\_go \wedge W\_go)$

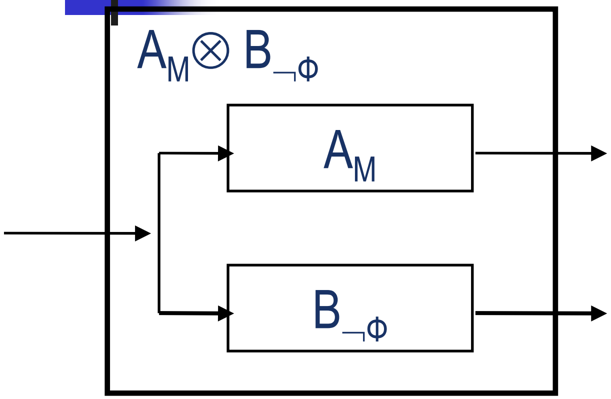
$$\neg\Psi = \mathbf{F}(N\_go \wedge W\_go)$$

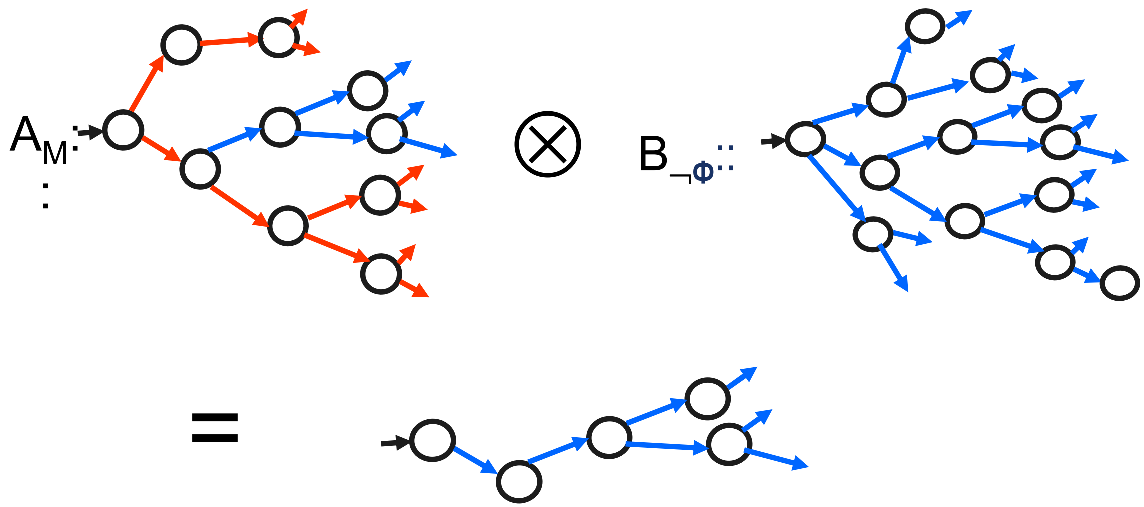


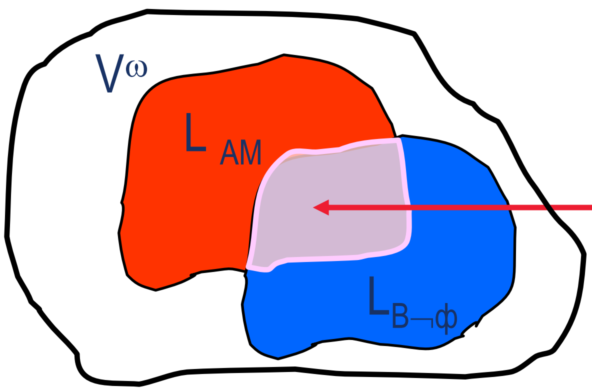


Если синхронная композиция $B_M \otimes B_{\neg\Psi}$ допускает цепочку, которая переводит автомат $B_{\neg\Psi}$ в состояние 1, то система управления некорректна

Model Checking для LTL: сложность







$$L_{AM} \cap L_{B\neg\Phi} = L_{AM\otimes B\neg\Phi}$$

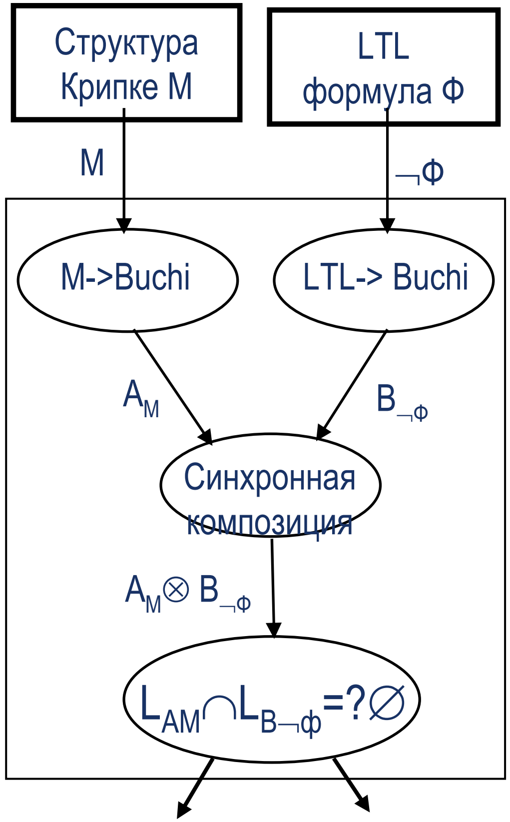
Сложность автомата Бюхи B_Φ экспоненциальна относительно сложности формулы Ф

Проверка пустоты пересечения линейна относительно сложности автомата Бюхи

Model Checking для LTL: Алгоритм

Общий алгоритм проверки выполнимости формулы LTL для структуры Крипке M:

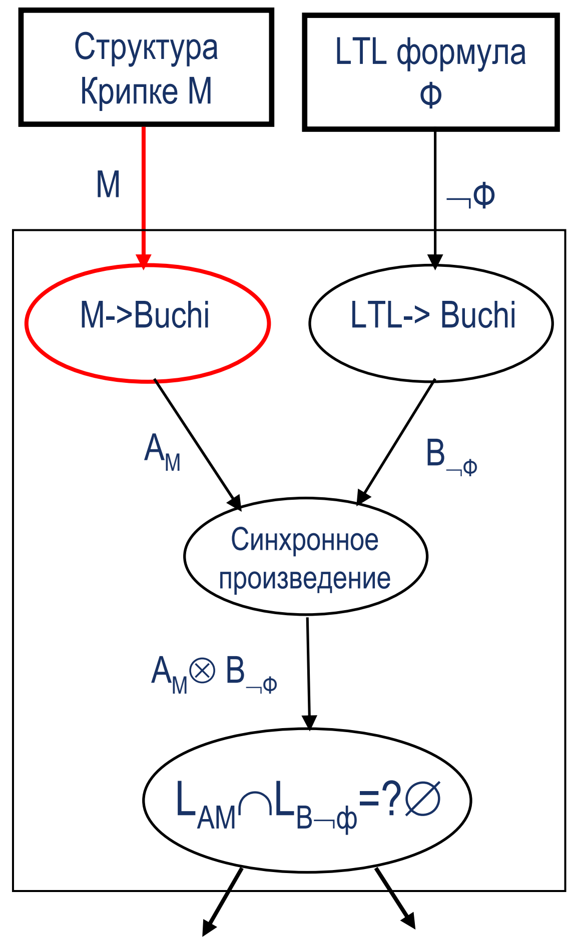
- По структуре Крипке M строится автомат Бюхи A_M. Этот автомат допускает все возможные траектории структуры M (цепочки подмножеств атомарных предикатов)
- По формуле LTL Ф строится автомат Бюхи $B_{\neg\Phi}$, допускающий множество вычислений, которые удовлетворяют $\neg\Phi$
- Строится автомат $A_M \otimes B_{\neg\Phi}$. Этот автомат – синхронная композиция автоматов A_M и $B_{\neg\Phi}$, допускает пересечение языков, допускаемых каждым компонентным автоматом
- Проверяется, допускает ли автомат $A_M \otimes B_{\neg\Phi}$ пустой язык
- Формула Ф выполняется для M, если и только если $A_M \otimes B_{\neg\Phi}$ допускает пустой язык



Все эти шаги представляют проблемы
Мы их рассмотрим последовательно

Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

- 1. Автоматы Бюхи, их формальное определение, примеры
- 2. Построение по структуре Крипке M такого автомата Бюхи A_M, который допускает все возможные вычисления структуры M
- 3. Синхронная композиция двух автоматов Бюхи. Обобщенный автомат Бюхи и его связь с обычным автоматом Бюхи
- 4. Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Ф автомата Бюхи $B_Ф$
- 5. Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи



Автоматы Бюхи: конечная модель задания омега языков

- Классический конечный автомат распознает только конечные цепочки, которые переводят его в заключительное состояние

- Для представления бесконечных вычислений используется автомат Бюхи (Richard Büchi) : $B = (Q, \Sigma, I, \delta, F)$:

  Q – конечное множество состояний;

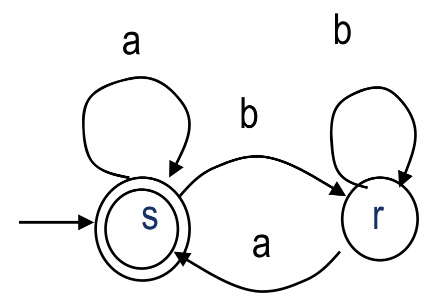
  Σ - конечное множество – входной алфавит

  $I \subseteq Q$ - конечное множество начальных состояний

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

  $F \subseteq Q$ – множество *принимающих* состояний

B::

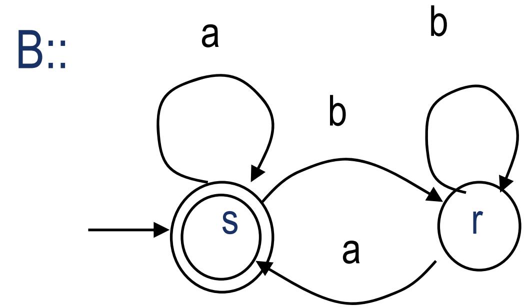


- Вычисление автомата B над ω-словом $w=a_0a_1\ldots \in \Sigma^\omega$ - это бесконечная цепочка $\rho=q_0q_1 \ldots$ такая, что $s_0 \in I$ и $(\forall i \in N)\ (q_i\, a_i\, q_{i+1}) \in \delta$

- ρ допускается, iff $(\exists q \in F)\ q_i=q$ для бесконечного числа $i \in N$ (т.е. $\inf(\rho) \cap F \neq \varnothing$)

- Язык $L_B \subseteq \Sigma^\omega$ - множество ω-слов, для которых $\exists$ допускаемое вычисление ρ.

Автомат Büchi vs конечный автомат



Допускаемый язык:

Как конечный автомат:

L_A= все (конечные) слова, кончающиеся на a, или

$(a^*(bb^*a)^*)^* = \varepsilon+(a+b)^*a$

Как автомат Бюхи:

$L_A = (b^*a)^\omega$ - все бесконечные цепочки с бесконечным числом вхождений a

Синтаксис совпадает, семантика различна

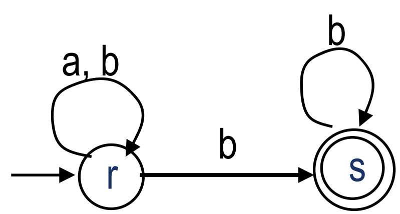
Недетерминированный автомат Бюхи

Для недетерминированного автомата Бюхи не всегда существует эквивалентный детерминированный автомат Бюхи

Пример: Когда-нибудь в будущем будет выполняться только b

FGb

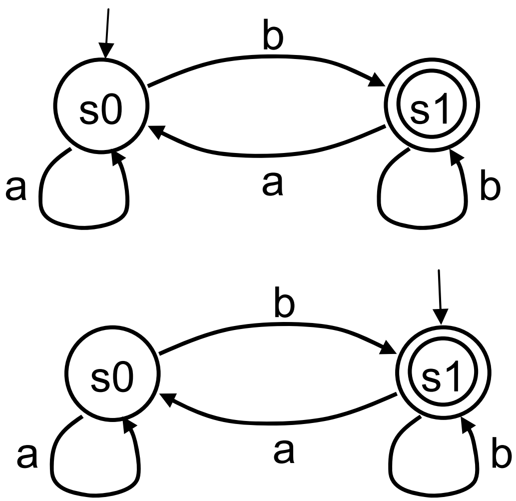
A::



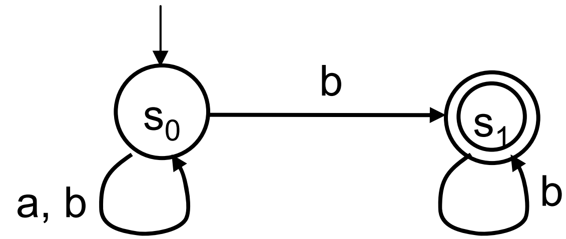
Не существует эквивалентного детерминированного автомата Бюхи

В отличие от КА, недетерминированные автоматы Бюхи допускают более широкий класс языков, чем детерминированные

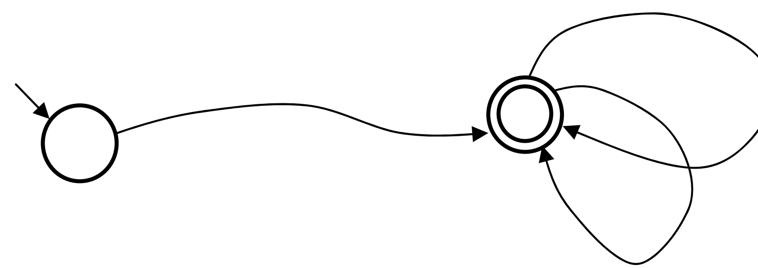
Автоматы Бюхи



Два эквивалентных автомата Бюхи, допускающих цепочки с бесконечным числом вхождений b



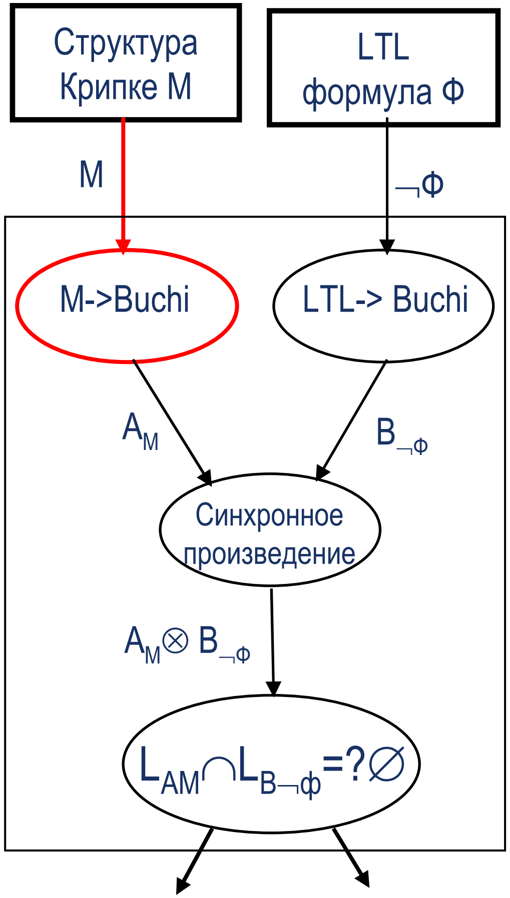
Автомат Бюхи, допускающий цепочки с конечным числом вхождений a и бесконечным числом вхождений b



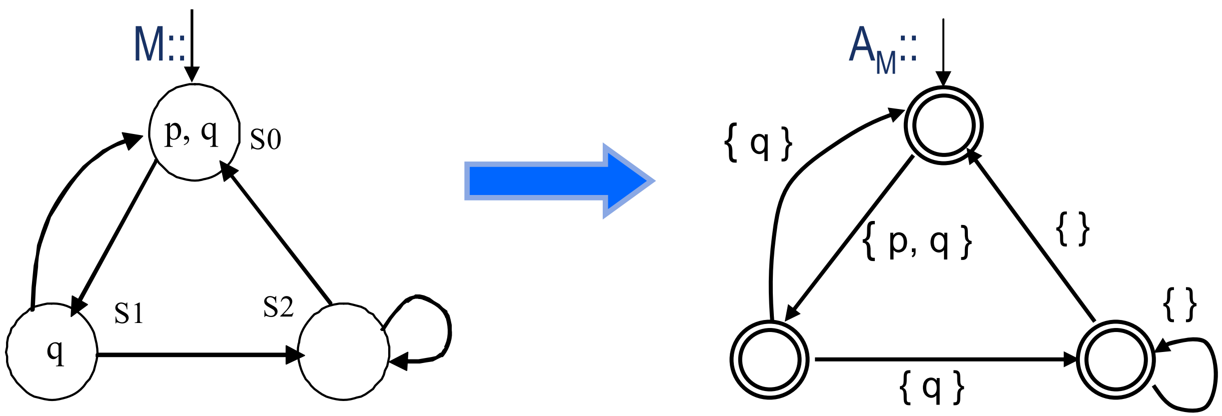
Допустимое вычисление автомата Бюхи всегда имеет цикл с включенным финальным состоянием

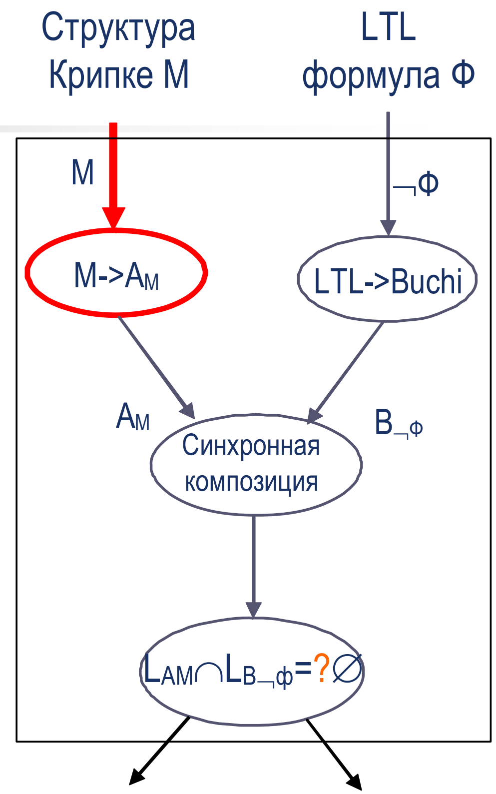
Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

- 1. Автоматы Бюхи, их формальное определение, примеры
- 2. Построение по структуре Крипке M такого автомата Бюхи A_M, который допускает все возможные вычисления структуры M
- 3. Синхронная композиция двух автоматов Бюхи. Обобщенный автомат Бюхи и его связь с обычным автоматом Бюхи
- 4. Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Φ автомата Бюхи B_Φ
- 5. Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи



Структура Крипке $\Rightarrow$ автомат Бюхи

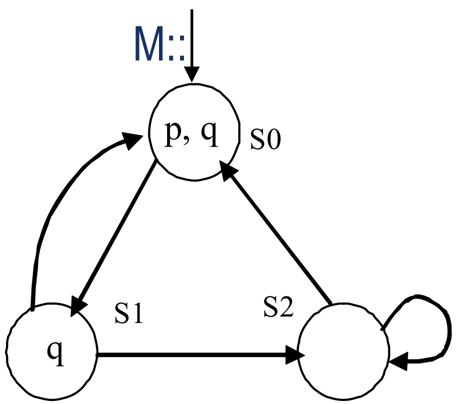


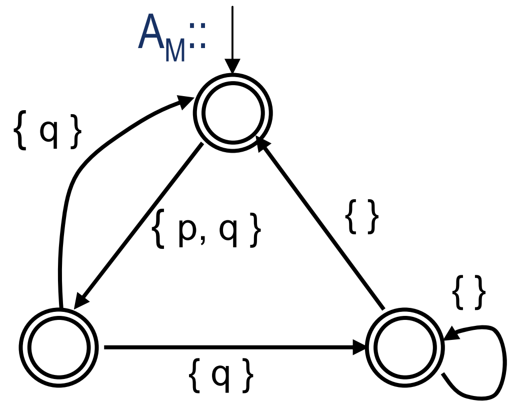


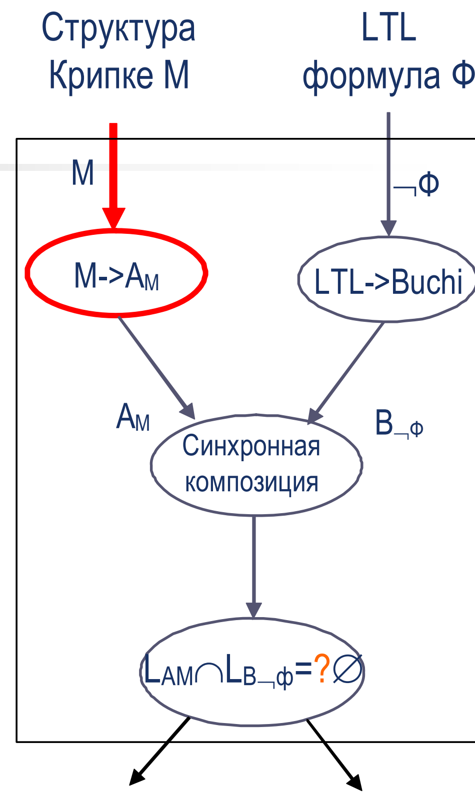
- Символами автомата Бюхи в нашей теории являются подмножества атомарных предикатов (то, что может стоять в состояниях структуры Крипке)
- Каждое состояние структуры Крипке M –> принимающее состояние автомата Бюхи A_M
- Каждая выходная стрелка в состоянии s автомата A_M помечается множеством L(s) – предикатами, истинными в s

Структуру Крипке M можно рассматривать как автомат Бюхи A_M, принимающий язык L_M над словарем 2^{AP}. Т.е. входной словарь автомата Бюхи – множество подмножеств структуры Крипке. Все состояния A_M – принимающие!

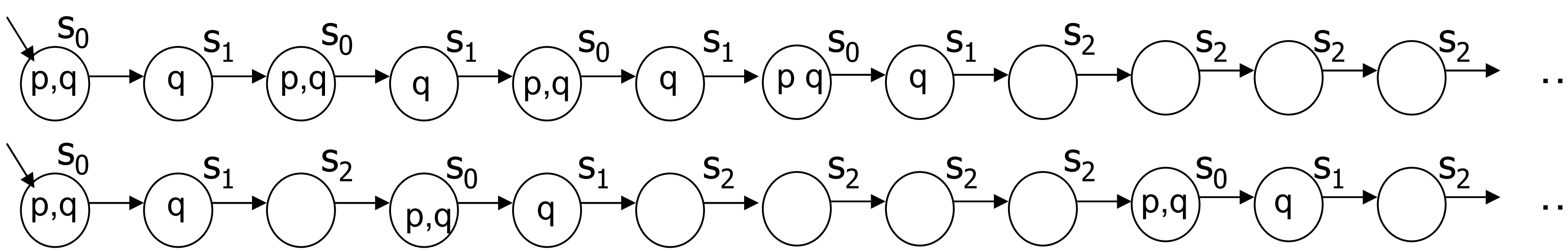
Структура Крипке ⇒ автомат Бюхи







Вычисления M:



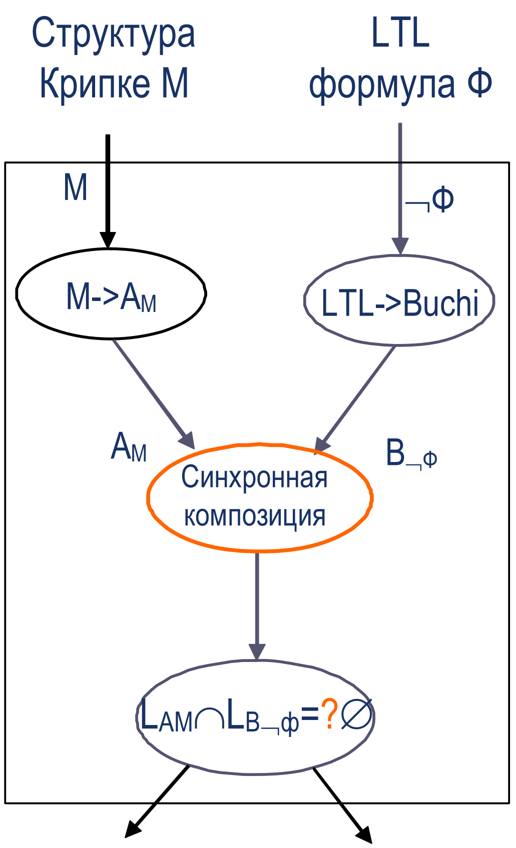
Цепочки языка, допускаемого A$_M$:

{p,q}{q}{p,q}{q}{p,q} {q} (p,q} {q} {}{}{}{} ...

{p,q}{q}{}{p,q} {q} { }{}{}{p,q}{q}{} {} ...

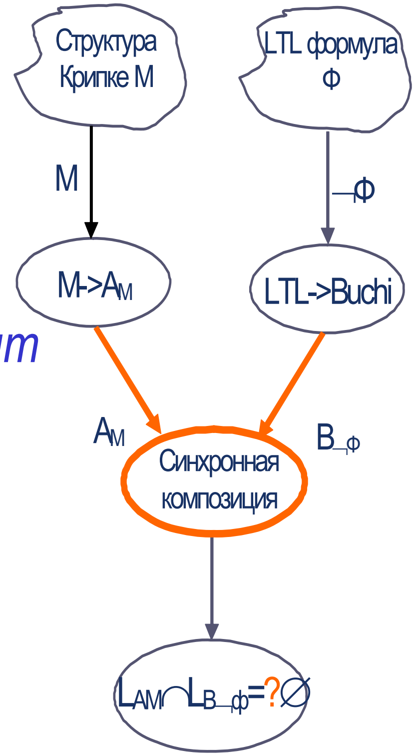
Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

- 1. Автоматы Бюхи, их формальное определение, примеры
- 2. Построение по структуре Крипке M такого автомата Бюхи A_M, который допускает все возможные вычисления структуры M
- **3. Синхронная композиция двух автоматов Бюхи. Обобщенный автомат Бюхи и его связь с обычным автоматом Бюхи**
- 4. Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Ф автомата Бюхи $B_Ф$
- 5. Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи



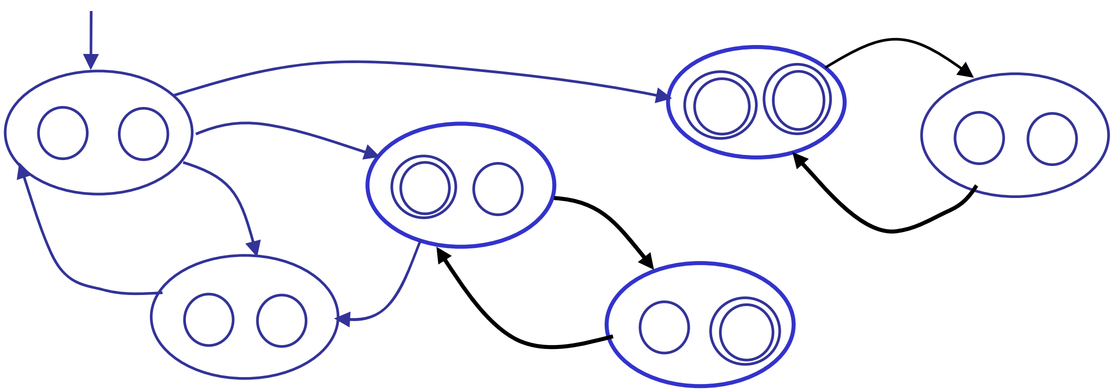
Синхронная композиция автоматов Бюхи дает обобщенный автомат Бюхи

Пусть $A=(S, \Sigma, I_A, \delta_A, F_A)$, $B=(Q, \Sigma, I_B, \delta_B, F_B)$ - автоматы Büchi



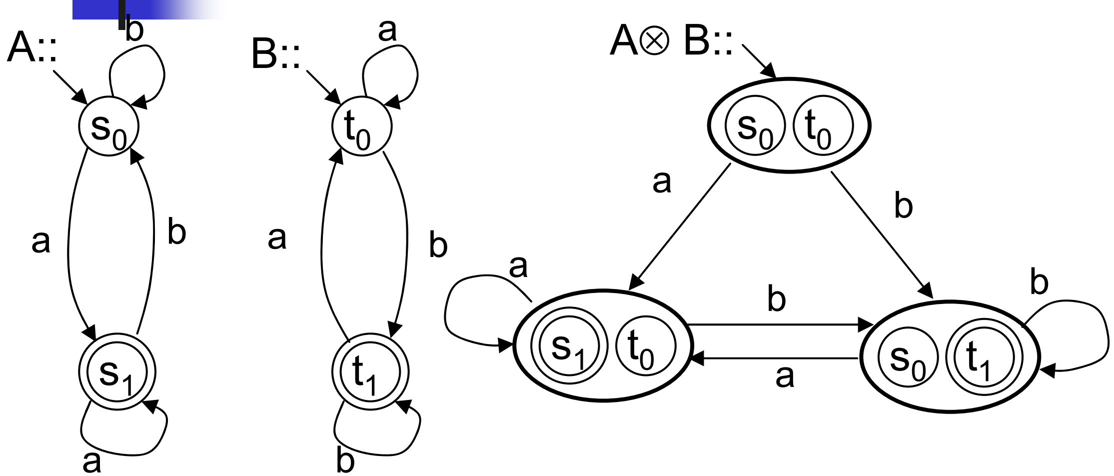
$A\otimes B=(S\times Q, \Sigma, I_{AB}, \delta_{AB}, F_{AB})$: $(s,q)\in I_{AB}$ iff $s\in I_A$ и $q\in I_B$
$\delta_{AB}((s,q),a) \in \delta_A(s,a) \times \delta_B(q,a)$

F_{AB} определяется так:
"цепочка принимается автоматом $A\otimes B$ ттогда, когда она проходит бесконечное число раз через такое состояние (s_1,q_1), что $s_1\in F_A$, и бесконечное число раз через такое состояние (s_2,q_2), что $q_2\in F_B$"



Поскольку необязательно, чтобы в композиции допускающие состояния одного автомата и допускающие состояния другого проходились одновременно, в результате композиции получаем обобщенный автомат Бюхи

Пример композиции автоматов Бюхи



Обобщенный автомат Бюхи имеет $\mathcal{F}=\{F_1, F_2, \ldots F_n\}$ – множество заключительных состояний

A допускает цепочки с бесконечным числом a

B допускает цепочки с бесконечным числом b

A ⊗ B допускает цепочки с бесконечным числом и a, и b

Обобщенный автомат Бюхи допускает цепочку σ, iff под воздействием σ автомат проходит состояния из каждого подмножества F_i бесконечное число раз

$F = \{ \{<s_1, t_0>\}, \{<s_0, t_1>\}\}$ – эти цепочки должны пройти бесконечное число раз и одно, и другое состояние

$$\boxed{L_{A \otimes B} = (aa^*bb^*)^{\omega}}$$

Обобщенный автомат Бюхи - пример

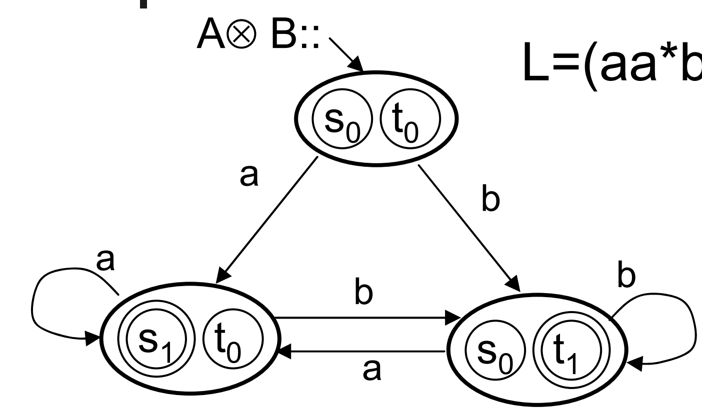
В автомате A⊗B заключительные состояния F = { {<t_1, s_0>}, {<t_0, s_1>}} – допускаемые цепочки должны пройти бесконечное число раз и одно, и другое состояние, т.е. и <t1, s0>, и <t0, s1>

A и B - автоматы Бюхи, допускающие цепочки с *конечным* числом вхождений одной буквы и бесконечным числом вхождений другой.

Их синхронная композиция допускает пустой язык

Теорема. *Для обобщенного автомата Бюхи существует эквивалентный ему обыкновенный автомат Бюхи*

От обобщенного автомата Бюхи к автомату Бюхи

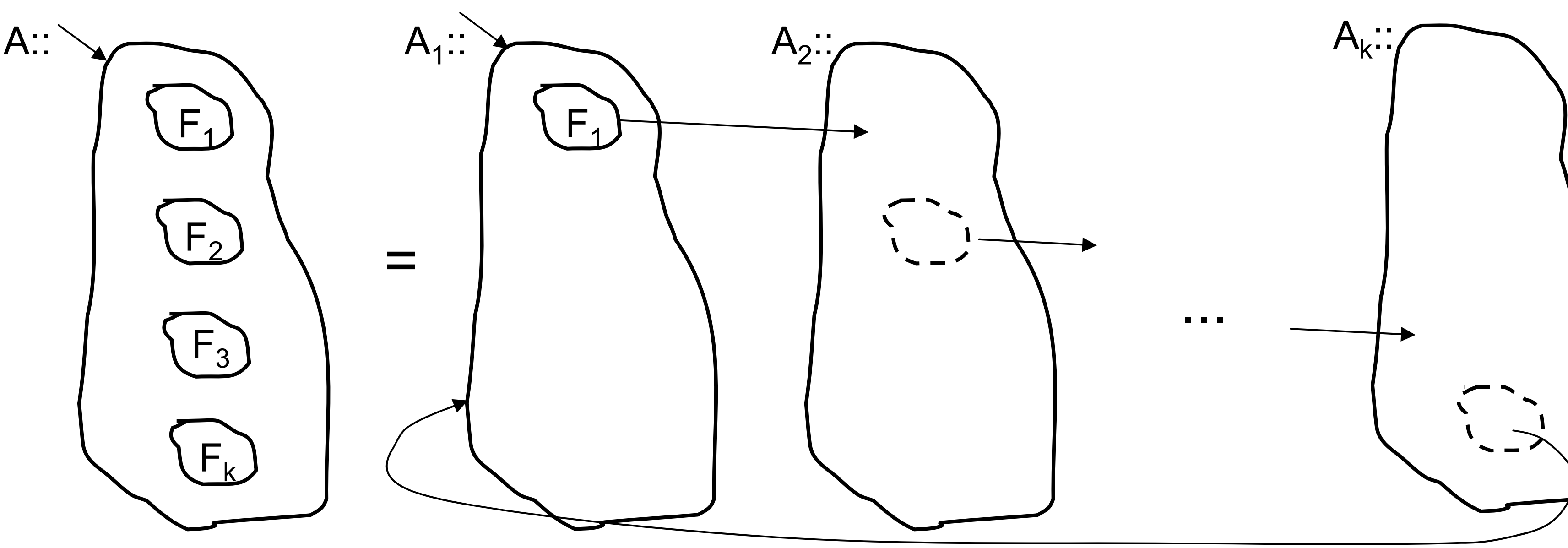


$L=(aa^*bb^*)^\omega$

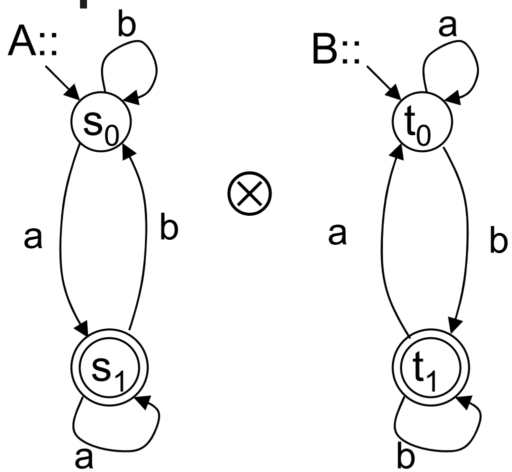
Обобщенный автомат Бюхи допускает цепочку σ, iff под воздействием σ автомат проходит состояния из каждого F_i бесконечное число раз

$F= \{F_1, F_2\} = \{ \{<s_1, t_0>\}, \{ <s_0, t_1> \} \}$

Существует простой алгоритм преобразования
Обобщенный автомат Бюхи –> Автомат Бюхи



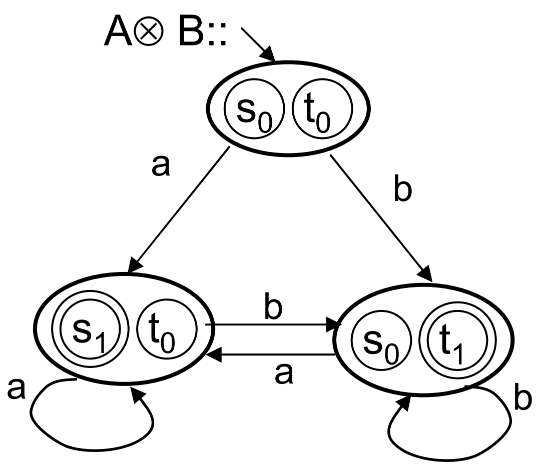
Синхронная композиция автоматов Бюхи



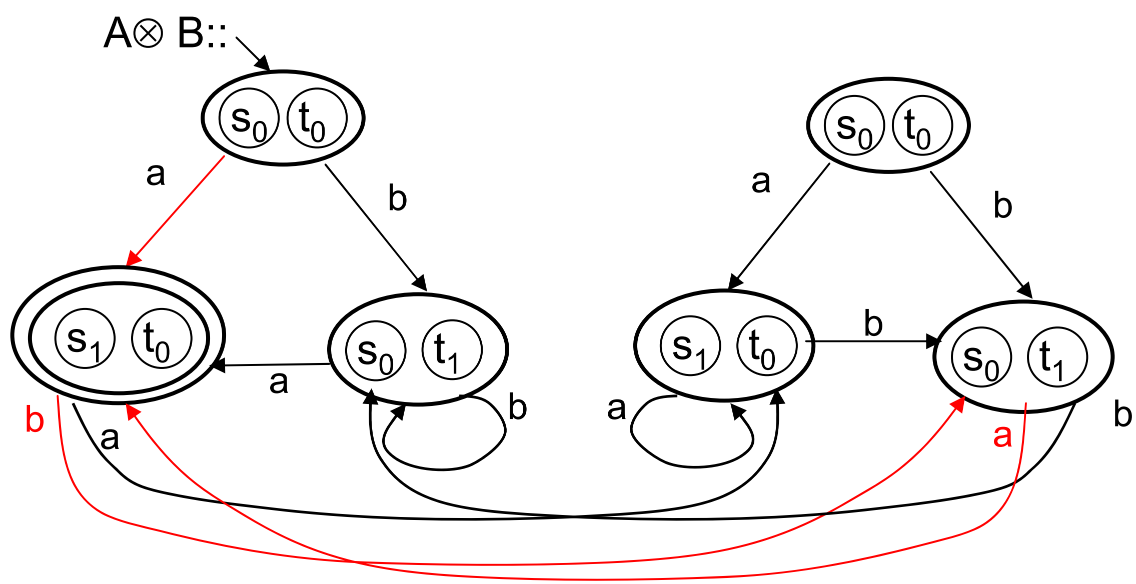
A допускает все цепочки с бесконечным числом a

B допускает все цепочки с бесконечным числом b

Их синхронная композиция A ⊗ B допускает все цепочки с бесконечным числом вхождений и a, и b

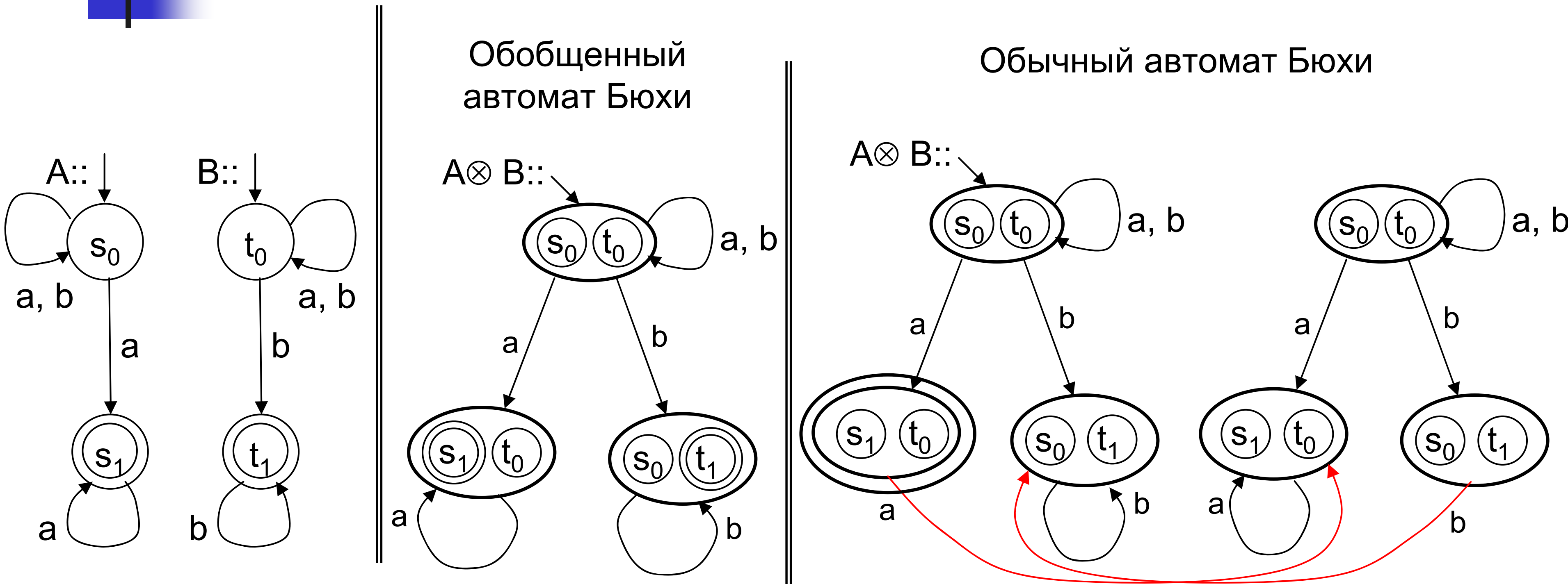


$L=(aa^*bb^*)^\omega$



Цикл ababab … проходит через допускающее состояние

Синхронная композиция автоматов Бюхи



Автомат А допускает язык с конечным числом b и бесконечным числом а

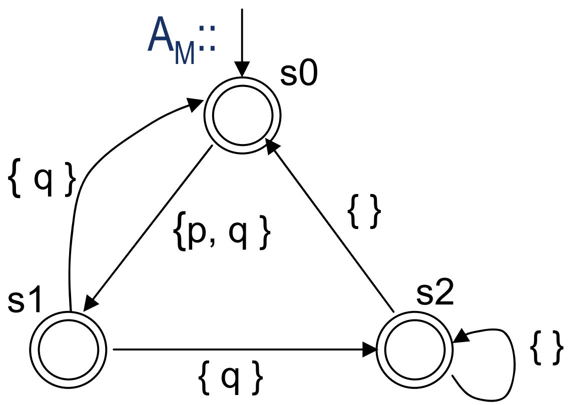
Автомат В допускает язык с конечным числом а и бесконечным числом b

Пересечение таких языков пусто (нет бесконечных цепочек из a и b, включающих конечное число и a, и b

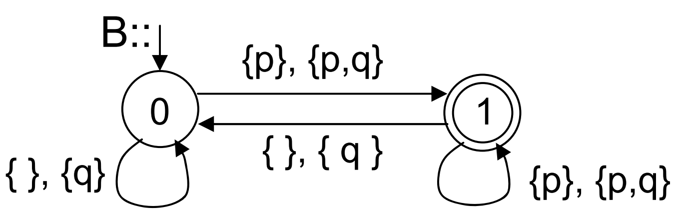
Синхронная композиция автоматов А и В дает автомат Бюхи, допускающий пустой язык – нет цикла, включающего допускающее состояние

Пример: композиция двух автоматов Бюхи

Автомат A получен из структуры Крипке



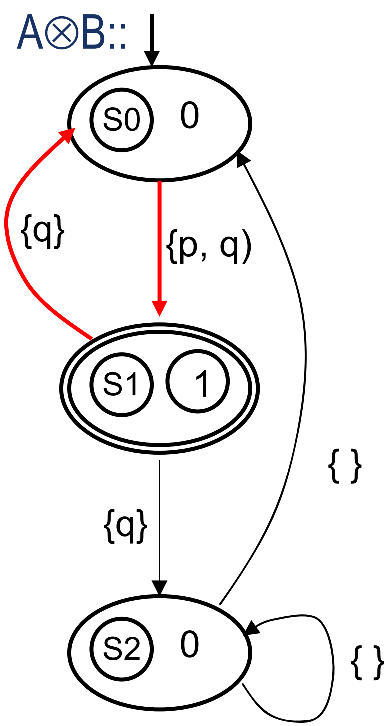
Автомат Бюхи B допускает все цепочки, удовлетворяющие свойству GFp



Проверим, выполняется ли на A_M требование $FG\neg p$

Строим отрицание формулы $FG\neg p$:

$$\neg FG\neg p = GFp$$



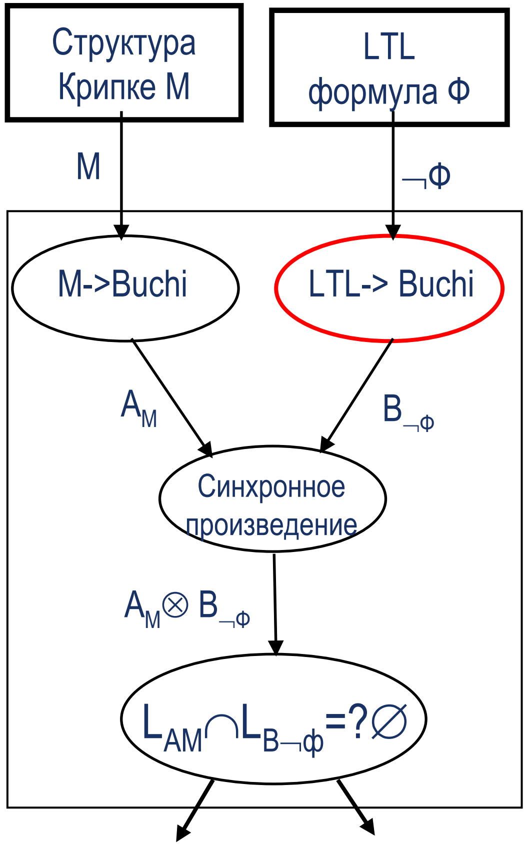
Автомат A⊗B допускает цепочку

{p,q} {q} {p,q} {q} {p,q} {q} {p,q} {q} {p,q} {q} ...

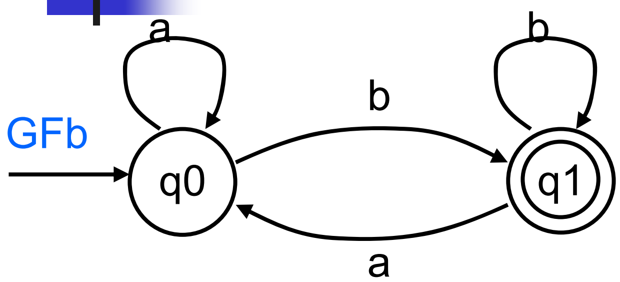
Поскольку у автомата A_M все состояния принимающие, проблема решается просто, без использования обобщенного автомата Бюхи

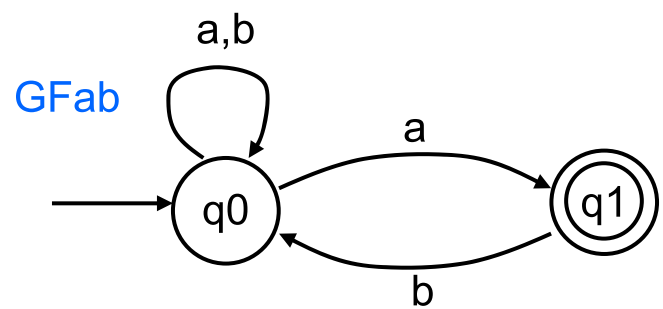
Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

- 1. Автоматы Бюхи, их формальное определение, примеры
- 2. Построение по структуре Крипке M такого автомата Бюхи A_M, который допускает все возможные вычисления структуры M
- 3. Синхронная композиция двух автоматов Бюхи. Обобщенный автомат Бюхи и его связь с обычным автоматом Бюхи
- **4. Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Φ автомата Бюхи $B_Φ$**
- 5. Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи



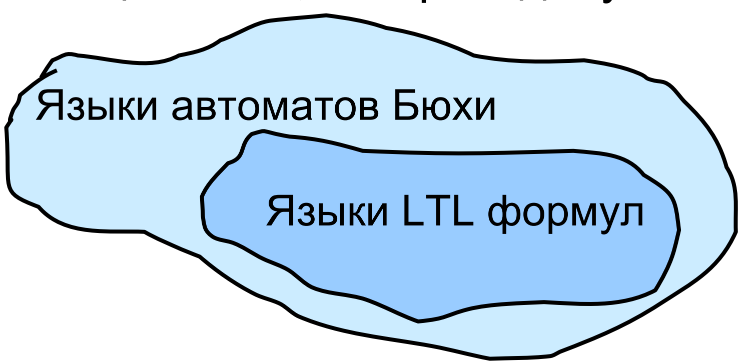
Автоматы Бюхи и LTL формулы

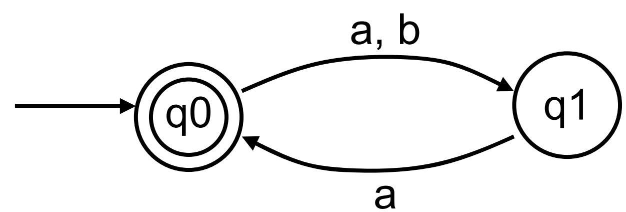




Теорема. Для каждой LTL формулы существует автомат Бюхи, который является моделью этой формулы (т.е.допускает тот же язык, который определяется LTL формулой)

Теорема. Существуют автоматы Бюхи, для которых не существует формулы LTL, описывающих язык, который допускается этими автоматами

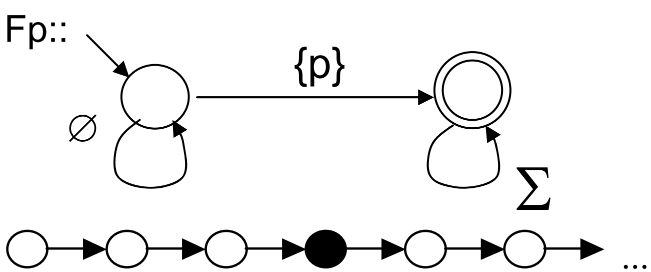


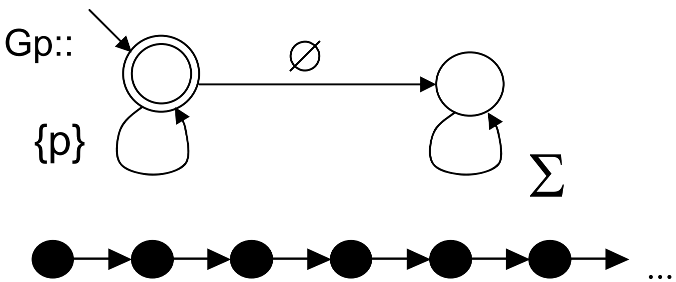


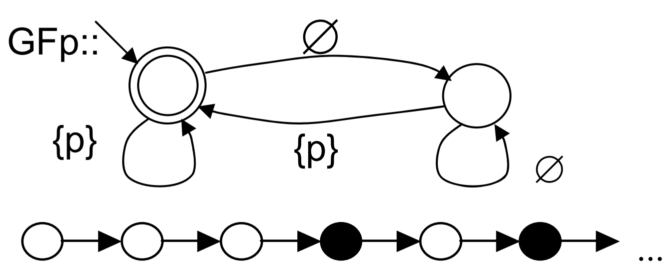
Пример: автомат Бюхи, допускающий цепочку, в которой ‘a’ встречается на каждом втором шаге: $((a+b)a)^{\omega}$. LTL формулы для такого множества цепочек не существует

Примеры автоматов Бюхи, допускающих цепочки, удовлетворяющие LTL формулам

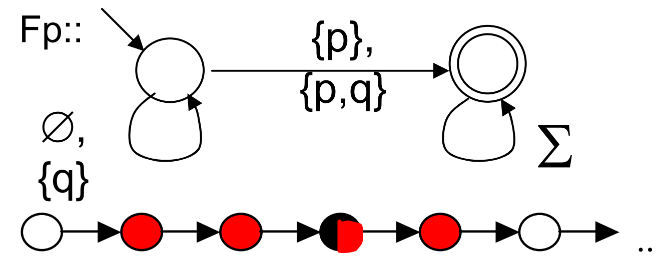
AP={p}; Σ={ ∅, {p} }

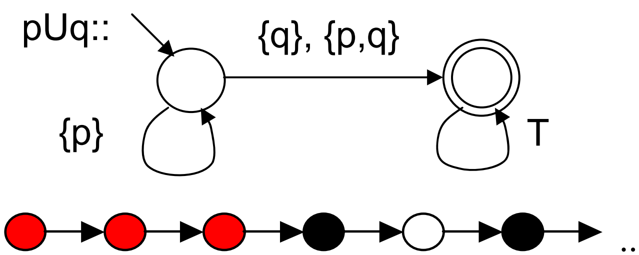


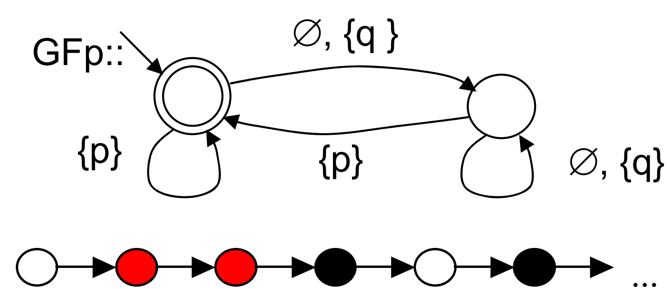




AP={p,q}; Σ={ ∅, {p}, {q}, {p,q} }

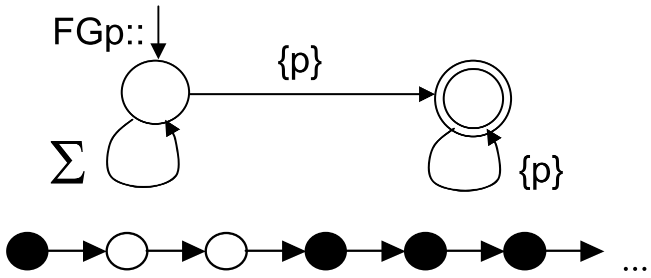




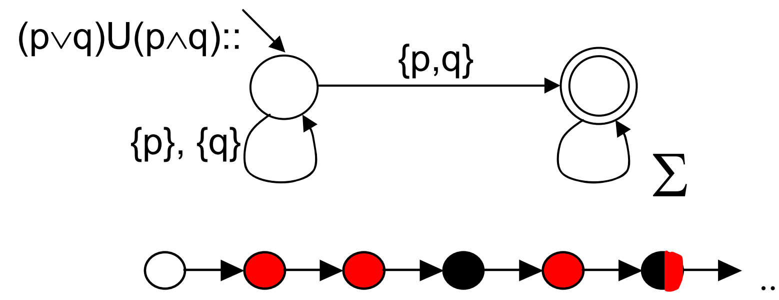


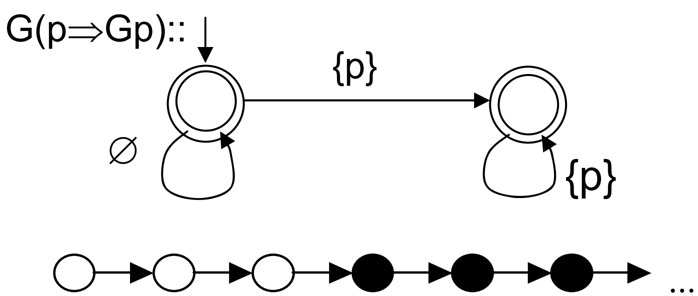
Примеры автоматов Бюхи, допускающих цепочки, удовлетворяющие LTL формулам

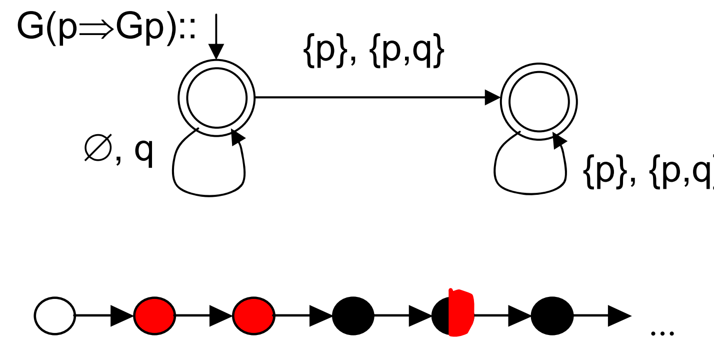
AP={p}; Σ={ ∅, {p} }



AP={p,q}; Σ={ ∅, {p}, {q}, {p,q} }





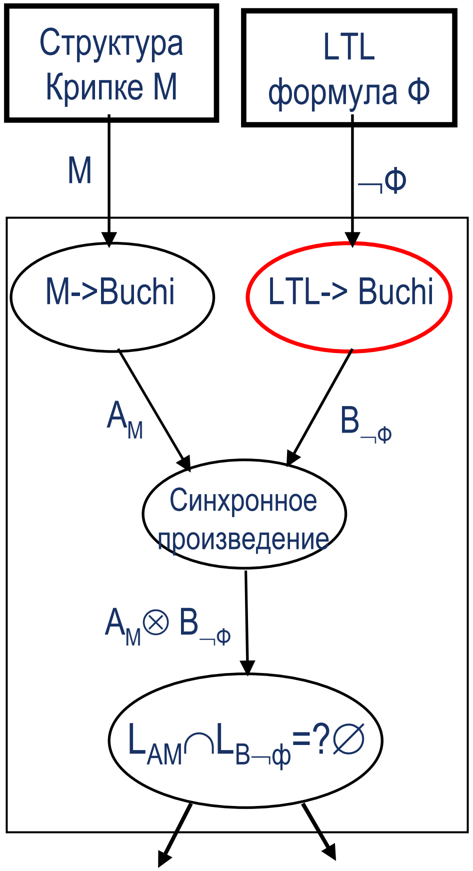


Формула LTL ⇒ автомат Бюхи

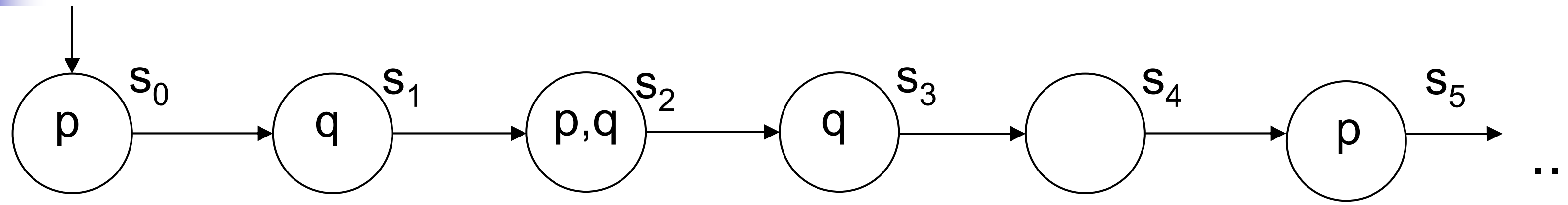
Существуют алгоритмы, которые по любой LTL-формуле φ строят автомат Бюхи B_φ, допускающий удовлетворяющие φ траектории.

Сложность B_φ в общем случае экспоненциальна: $2^{|\varphi|}$

Из-за того, что обычно формулы малы, такие алгоритмы приемлемы

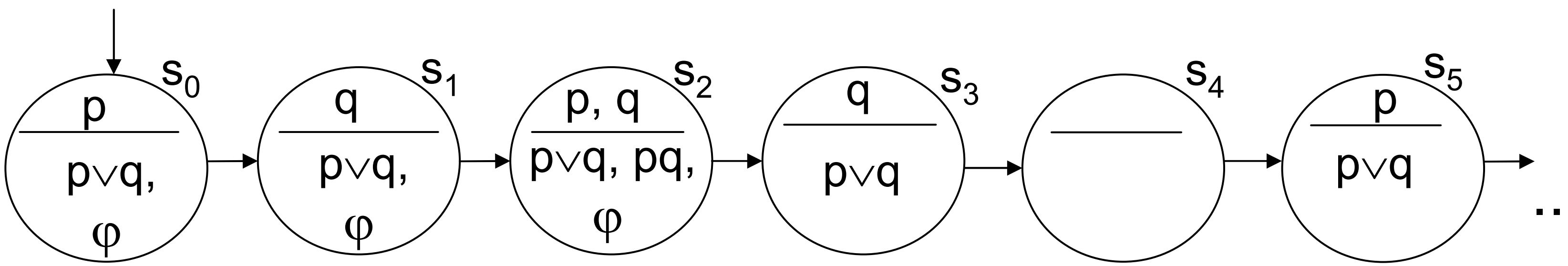


Автомат Бюхи по формуле LTL: Замыкание формулы



Вычисление, на котором хотим проверить выполнение формулы $\varphi = (p \vee q)\mathbf{U}(p \wedge q)$

b)



В каждое состояние S вычисления включаем множество подформул формулы φ, которые выполняются в вычислении, начинающемся из S

Замыкание формулы φ - это множество подформул этой формулы

$$cl(\,(p \vee q)\mathbf{U}(p \wedge q)\,) = \{\ p,\ q,\ p \vee q,\ p \wedge q,\ (p \vee q)\mathbf{U}(p \wedge q)\ \}$$

Автомат Бюхи по формуле LTL: Атомы формулы

Атом формулы ϕ *:*
максимальное непротиворечивое подмножество $cl(\phi)$, которое может помечать состояния вычислений, на которых выполняется ϕ

Пусть грамматика LTL такова: $\phi ::= p \mid \phi 1 \vee \phi 2 \mid \neg\phi \mid X\phi \mid \phi_1 \; U \; \phi_2 \mid F\phi \mid G\,\phi$

Локальные правила построения атомов формулы Ф:

R1. $\psi \in A \;\Leftrightarrow\; \neg\psi \notin A$

R2. $\varphi_1 \vee \varphi_2 \in A \Leftrightarrow \varphi_1 \in A$ или $\varphi_2 \in A$

R3. $\varphi_1 \mathbf{U} \varphi_2 \in A$ и $\varphi_2 \notin A \Rightarrow \varphi_1 \in A$

R4. $\varphi_2 \in A \Rightarrow \varphi_1 \mathbf{U} \varphi_2 \in A$

Для выводимых операторов $\wedge$, $\Rightarrow$, F и G правила можно дополнить:

R5. $\varphi 1 \wedge \varphi 2 \in A \Leftrightarrow \varphi 1 \in A$ и $\varphi 2 \in A$

R6. $\varphi 1 \Rightarrow \varphi 2 \in A \Leftrightarrow \neg\varphi 1 \in A$ или $\varphi 2 \in A$

R7. $\psi \in A \quad \Rightarrow \quad \mathbf{F}\psi \in A$

R8. $\mathbf{G}\psi \in A \quad \Rightarrow \quad \psi \in A$

Атомы составляют множество состояний искомого автомата Бюхи

Примеры построения атомов LTL формул

Формула Fp:

Замыкание cl(Fp) формулы Fp содержит две подформулы: { p, Fp }.
У этого множества подформул всего 4 подмножества:
$\varnothing$, { p }, { Fp }, { p, Fp }.

Но множество { p } противоречит правилу R7: $\psi \in A \Rightarrow \mathbf{F}\psi \in A$

A1= $\varnothing$
A2= { **F**p }
A3= { p, **F**p}

Формула GFp:

Замыкание cl(GFp) содержит три подформулы: {p,Fp,GFp}.
У этого множества 8 подмножеств:
$\varnothing$, { p }, { Fp }, { GFp }, { p, Fp }, {p, GFp}, { Fp, GFp }, { p, Fp, GFp }

Множество { p } противоречит правилу R7: $\psi \in A \Rightarrow \mathbf{F}\psi \in A$

Множества {GFp} и {p, GFp } противоречат R8: $\mathbf{G}\psi \in A \Rightarrow \psi \in A$

A1= $\varnothing$
A2= { **F**p }
A3= {p, **F**p}
A4= { **F**p, **GF**p }
A5= {p, **F**p, **GF**p}

Формула pUq:

Замыкание cl(pUq) содержит три подформулы: {p, q, pUq }.
У этого множества 8 подмножеств:
$\varnothing$, { p }, { q }, { pUq }, { p, q }, {p, pUq}, { q, pUq }, { p, q, pUq }

Множества { q } и { p, q } противоречат R4: $\varphi_2 \in A \Rightarrow \varphi_1 \mathbf{U} \varphi_2 \in A$

Множество { pUq } противоречит R3. $\varphi_1 \mathbf{U} \varphi_2 \in A$ и $\varphi_2 \notin A \Rightarrow \varphi_1 \in A$

A1= $\varnothing$,
A2= {p },
A3= {p, p**U**q },
A4= {q, p**U**q},
A5= {p, q, p**U**q}.

Построение атомов LTL формул: общее правило

1. Для каждой подформулы ψ замыкания cl(Ф) формулы Ф введем булеву переменную $x\psi$, такую, что $x_\psi \equiv \psi \in A$.

Например, пусть $\Phi = G(p \Rightarrow Gp)$, $cl(G(p \Rightarrow Gp)) = \{ p, Gp, p \Rightarrow Gp, G(p \Rightarrow Gp) \}$

Введем: $x = p \in A$; $y = Gp \in A$; $z = p \Rightarrow Gp \in A$; $v = G(p \Rightarrow Gp) \in A$

2. Выпишем те правила R1 – R8, в которых встречаются подформулы из cl(Ф).

Например, для нашей формулы это

R6. $\varphi_1 \Rightarrow \varphi_2 \in A \Leftrightarrow \neg\varphi_1 \in A$ или $\varphi_2 \in A$

R8. $G\psi \in A \quad \Rightarrow \quad \psi \in A$

3. Эти правила перепишем, как функции над введенными переменными:

Из R6: $f1 = z \Leftrightarrow \neg x \vee y$. Из R8: $f2 = y \Rightarrow x$ и $f3 = v \Rightarrow z$.

4. Выполнение в каждом атоме всех ограничений - конъюнкция этих функций: $f1 \wedge f2 \wedge f3$.

По таблице истинности находим, что $f1 \wedge f2 \wedge f3$ принимает истинное значение на наборах:

<x=Л, y=Л, z=И, v = Л>; следовательно,	A1= { p ⇒**G**p },
<x=Л, y=Л, z=И, v = И>;	A2= { p ⇒**G**p, **G**(p ⇒**G**p) },
<x=И, y=Л, z=Л, v = Л>;	A3= { p },
<x=И, y=И, z=И, v = Л>;	A4= {p, **G**p, p ⇒**G**p},
<x=И, y=И, z=И, v = И>;	A5= {p, **G**p, p ⇒**G**p, **G**(p ⇒**G**p) } .

Переходы между атомами: выполнение обязательств

Обязательство: ограничение на переходы из данного состояния, определяемого набором подформул, которые в этом состоянии должны выполниться. Например, если X$p\in$A, то из состояния A могут быть переходы только в состояния, в которых включено p

C1. $\mathbf{X}\psi \in A \quad\Leftrightarrow\quad \psi\in A'$

C2. $\varphi 1\mathbf{U}\varphi 2\in A$ *и* $\varphi 2\notin A \Rightarrow \varphi 1\mathbf{U}\varphi 2\in A'$

C3. $\varphi 1\mathbf{U}\varphi 2\in A'$ *и* $\varphi 1\in A \Rightarrow \varphi 1\mathbf{U}\varphi 2\in A$

Дополнительно, для выводимых операторов F и G:

C4. $\mathbf{F}\psi\in A$ *и* $\psi\notin A \Rightarrow \mathbf{F}\psi\in A'$

C5. $\mathbf{F}\psi\in A' \quad\Rightarrow\quad \mathbf{F}\psi\in A$

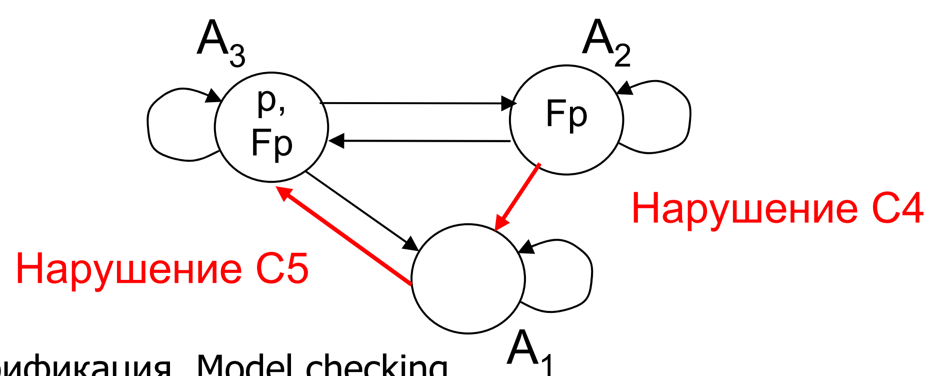
C6. $\mathbf{G}\psi\in A \quad\Leftrightarrow\quad \mathbf{G}\psi\in A'$ *и* $\psi\in A$

Пример: возможные переходы между атомами формулы Fp:

A1= ∅

A2= { **F**p }

A3= { p, **F**p}



Обобщенный автомат Бюхи по формуле LTL

1. Множество состояний $B_Ф$ – это множество атомов формулы Ф
2. Алфавит – множество подмножеств атомарных предикатов формулы Ф
3. Множество переходов – все переходы, не нарушающие обязательств С1-С6. Переход из каждого атома помечается подмножеством атомарных предикатов, содержащихся в атоме
4. Начальными состояниями будут те атомы, которые включают саму формулу Ф
5. Для каждой подформулы fi=φiUψi множество принимающих состояний Fi включает все такие состояния A, в которые или не входит fi, или входит ψi
   Для каждой подформулы fi=Fψ множество Fi включает все такие состояния A, в которые или не входит fi, или входит ψi
   Для каждой подформулы fi=Gψ множество Fi включает все такие состояния A, в которые или входит fi, или не входит ψ

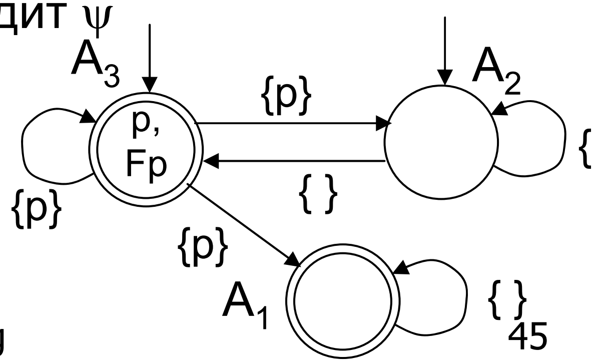
Пример: Формула Fp

A1= ∅

A2= { **F**p }

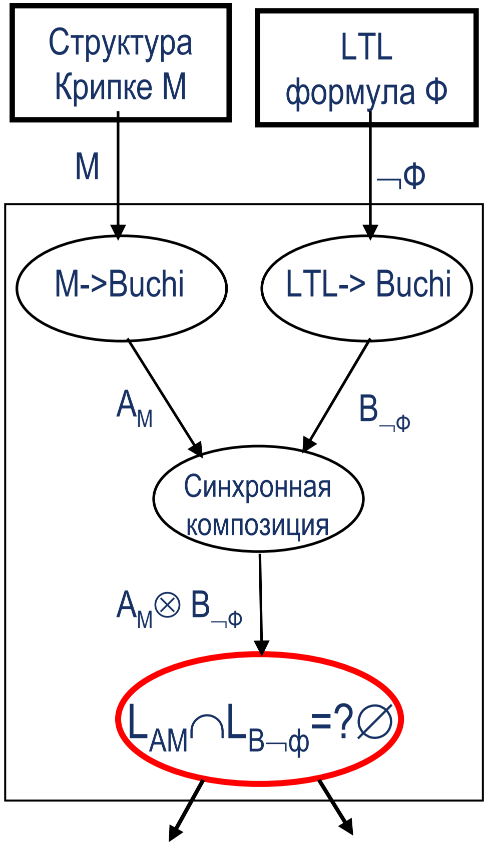
A3= { p, **F**p}

Автомат Бюхи:

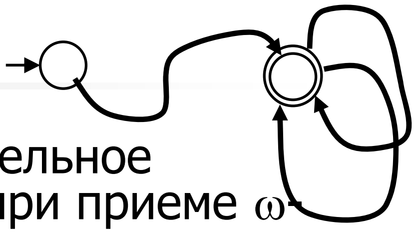


Проблемы, возникающие при выполнении алгоритма Model Checking для LTL

- 1. Автоматы Бюхи, их формальное определение, примеры
- 2. Построение по структуре Крипке M такого автомата Бюхи A_M, который допускает все возможные вычисления структуры M
- 3. Синхронная композиция двух автоматов Бюхи. Обобщенный автомат Бюхи и его связь с обычным автоматом Бюхи
- 4. Автомат Бюхи и формулы LTL. Алгоритм построения по формуле Φ автомата Бюхи B_Φ
- 5. Алгоритм проверки пустоты языка, допускаемого автоматом Бюхи



Автомат Бюхи и распознавание языка



- σ допускается автоматом Бюхи A, iff существует заключительное состояние A, которое проходится бесконечное число раз при приеме ω-цепочки σ
- Язык, допускаемый автоматом Бюхи, называется ω-регулярным языком
  - язык $L_\omega(A)$-*регулярный*, потому что он допускается конечным автоматом.
  - язык $L_\omega(A)$ называется *ω-языком*, потому что он содержит, в отличие от обычных языков, только бесконечные цепочки
- Поскольку множество заключительных состояний конечно, то если A допускает ω-цепочку σ, то он должен проходить заключительные состояния неопределенно часто. Следовательно, для того, чтобы A допускал хотя бы одну цепочку, в графе переходов A должен быть цикл, включающий одно из заключительных состояний по крайней мере один раз

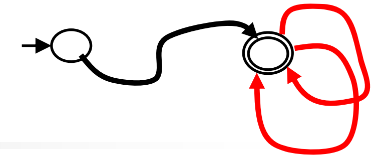
Проблема пустоты языка, допускаемого автоматом Бюхи, разрешима

Проверка пустоты языка, распознаваемого Автоматом Бюхи

- Как проверить, что язык, который допускает автомат Бюхи A, непустой?
- ***Теорема***. Автомат Бюхи A допускает непустую цепочку ТТОГДА, когда A содержит достижимую из начального состояния ССК, включающую хотя бы одно заключительное состояние
  - *НЕОБХОДИМОСТЬ*. Пусть A допускает ρ. Тогда, поскольку ρ ω-цепочка, в A существует достижимое заключительное состояние s, которое проходится под воздействием ρ бесконечное число раз. Все состояния A, которые проходятся в A под воздействием ρ после первого посещения s, образуют достижимую ССК.
  - *ДОСТАТОЧНОСТЬ*. Пусть A содержит ССК R, достижимую из начального состояния, и некоторое заключительное состояние $s \in R$. Тогда это s может быть достигнуто бесконечное число раз в A
- Алгоритм проверки наличия ССК на конечном множестве состояний линеен

ССК – сильно связанная компонента, любой узел достижим из любого узла

Контрпримеры при LTL Model Checking

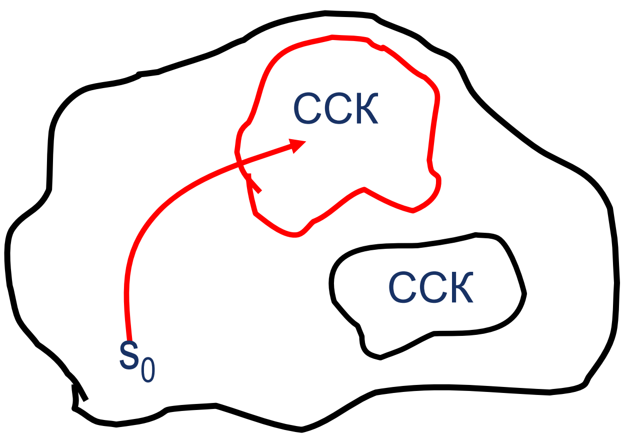


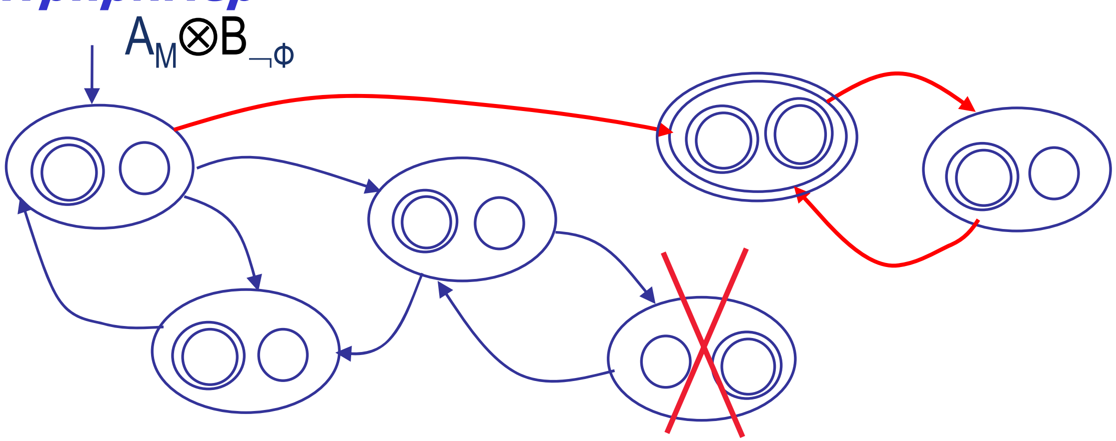
Проблема пустоты для автомата Бюхи разрешима:

1. Находим все Сильно Связные Компоненты графа переходов автомата
2. Оставляем те ССК, в которых есть хотя бы одно допускающее состояние
3. Проверяем, есть ли путь из начального состояния 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***





Таких у нас нет: все s в Ак - принимающие

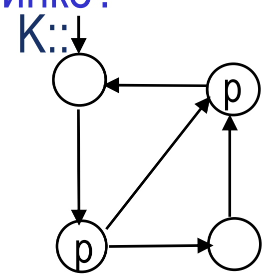
Пример: Model Checking для LTL

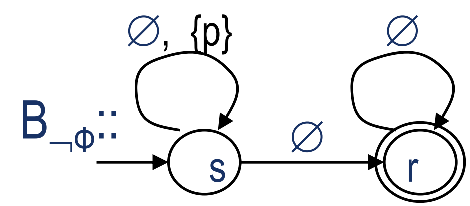
Удовлетворяет ли формуле GFp структура Крипке?

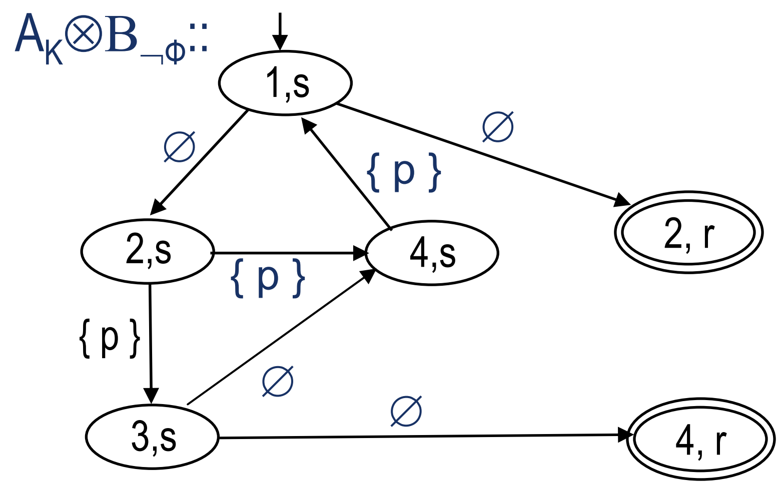
Φ = GFp истинна на вычислении, если неопределенно часто на нем встречаются p, разделенные конечным числом $\varnothing$. Это можно описать так: $L_\Phi = (\varnothing^* \ p)^\omega$

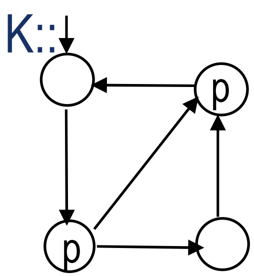
Если вычисление не удовлетворяет GFp, то оно удовлетворяет $\neg$GFp = FG $\neg$p = FG $\varnothing$, и его можно описать так: $L_{\neg\Phi} = (\varnothing + p)^* \ \varnothing^{\ \omega}$

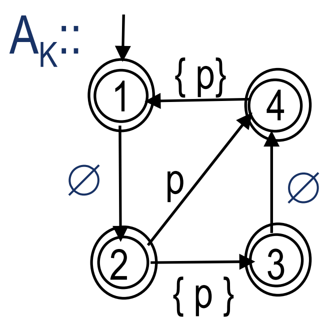
$\neg\Phi$ = FG $\varnothing$









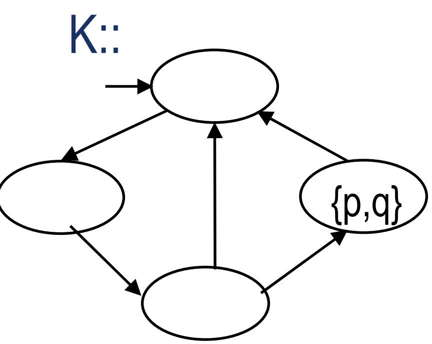


Поскольку нет достижимого цикла, включающего принимающее состояние, язык $L_{A_K \otimes B_{\neg\Phi}}$ пуст. Следовательно, на структуре Крипке выполняется формула GFp

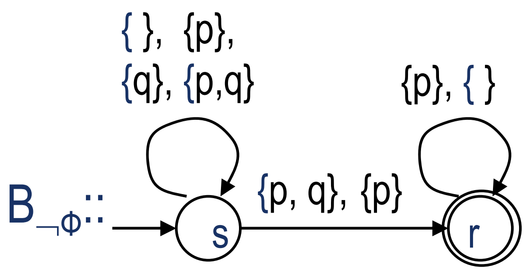
Model Checking для LTL: Пример

Проверим, выполняется ли формула Ф=G(p ⇒XFq) на структуре Крипке K?

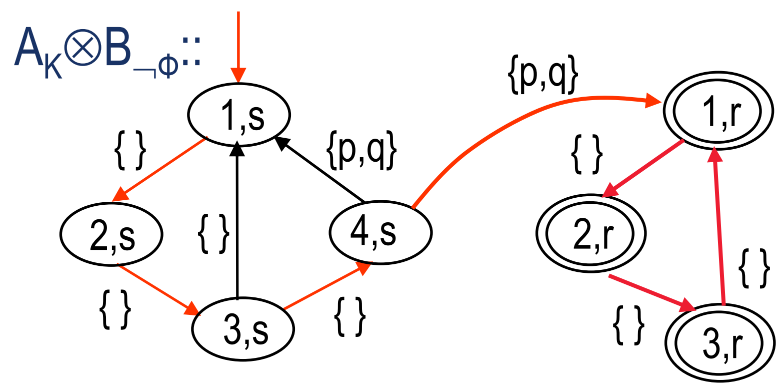
Отрицание Ф: $\neg\Phi=\neg G(p\Rightarrow XFq)= F(\, p \wedge XG\neg q)$



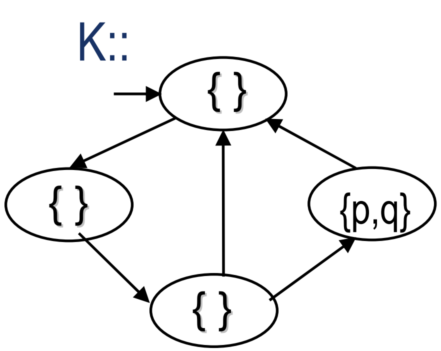
Автомат Бюхи $B_{\neg\Phi}$



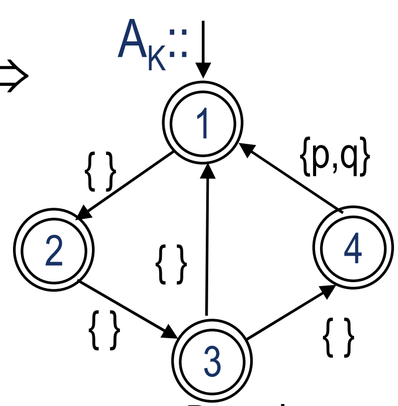
Композиция автоматов Бюхи:



Структура Крипке K ⇒ автомат Бюхи A_K



⇒



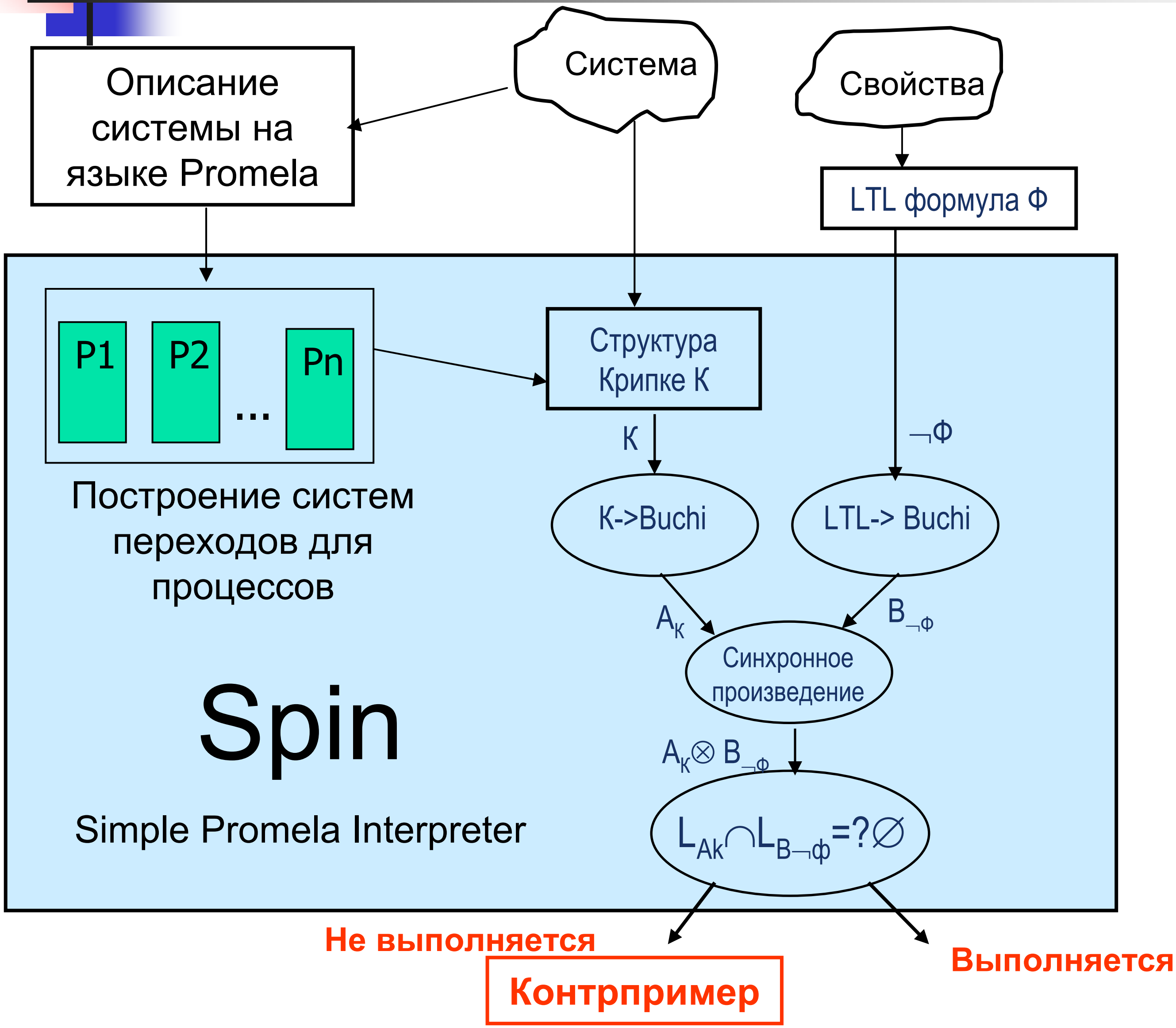
Есть цикл, включающий принимающее состояние ⇒ язык $L_{AK\otimes B\neg\Phi}$ непуст.

К не удовлетворяет Ф.

Контрпример: $1234(123)^{\omega}$

Система верификации Spin

ACM System Software Award 2001



Вход в систему Spin:

1. Модель взаимодействующих процессов на Promela
2. Атомарные предикаты, выражающие интересующие нас базовые свойства состояний
3. Проверяемое свойство в виде формулы PLTL

Пример

- В работе:

J.Penix et.al. Verification of time partitioning in DEOS Schedule kernel. Proc. 22rd Int conf on Software Engineering, 2000

приведен пример верификации реального планировщика процессов с помощью Spin'a.

Была найдена ошибка.

Система выдала контрпример длиной 2700 шагов (*не так просто определить эту тонкую ошибку*)

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!
- Возможность получения в результате выполнения алгоритма model checking контрпримера – траектории, на которой НЕ выполняется проверяемое свойство, 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

Удивительная красота метода model checking для формул LTL состоит в том, что тонкие абстрактные модели (ω-языки, автоматы Бюхи, их синхронная композиция, ...), которые и понять трудно, и реализовать нельзя, позволили разработать алгоритмы проверки свойств поведения реальных сложных технических систем: коммуникационных протоколов, бортовых систем космических аппаратов и т.п.

Спасибо за внимание