Методы верификации программ и схем

ЛЕКТОР:
Владимир Анатольевич Захаров

zakh@cs.msu.su

Лекция 4.

Свойства вычислений программ

Классификация свойств вычислений программ

Ограничения справедливости

Логика деревьев вычислений CTL*

Темпоральные логики CTL и LTL

Свойства вычислений программ

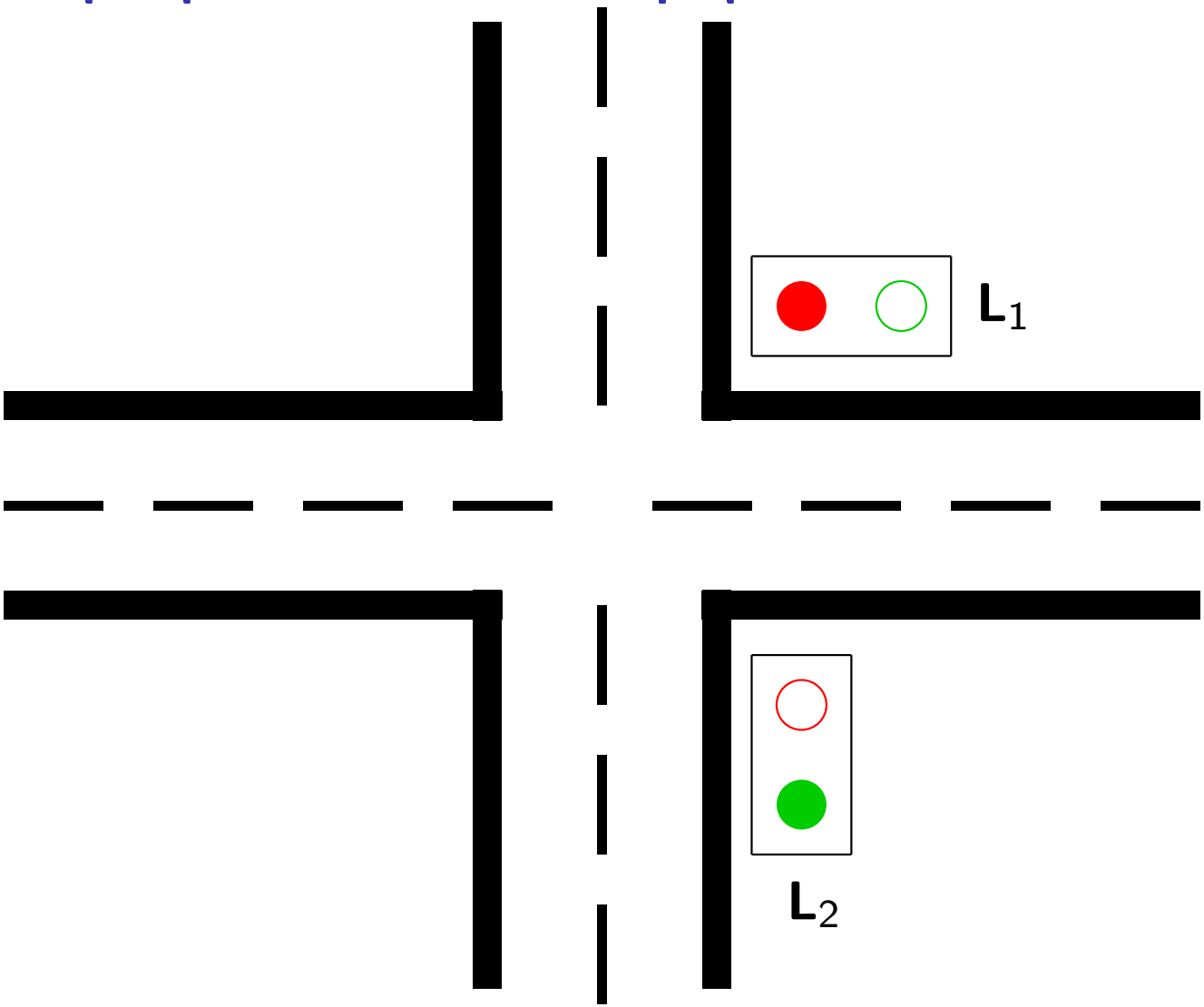
Вычисление реагирующей системы (управляющей программы, протокола, схемы) сопровождается осуществлением событий.

Многие реагирующие системы проектируют так, чтобы их вычисления вообще никогда не завершались.

Поэтому языки спецификаций реагирующих систем должны позволять описывать бесконечные последовательности событий, происходящие по ходу вычисления системы.

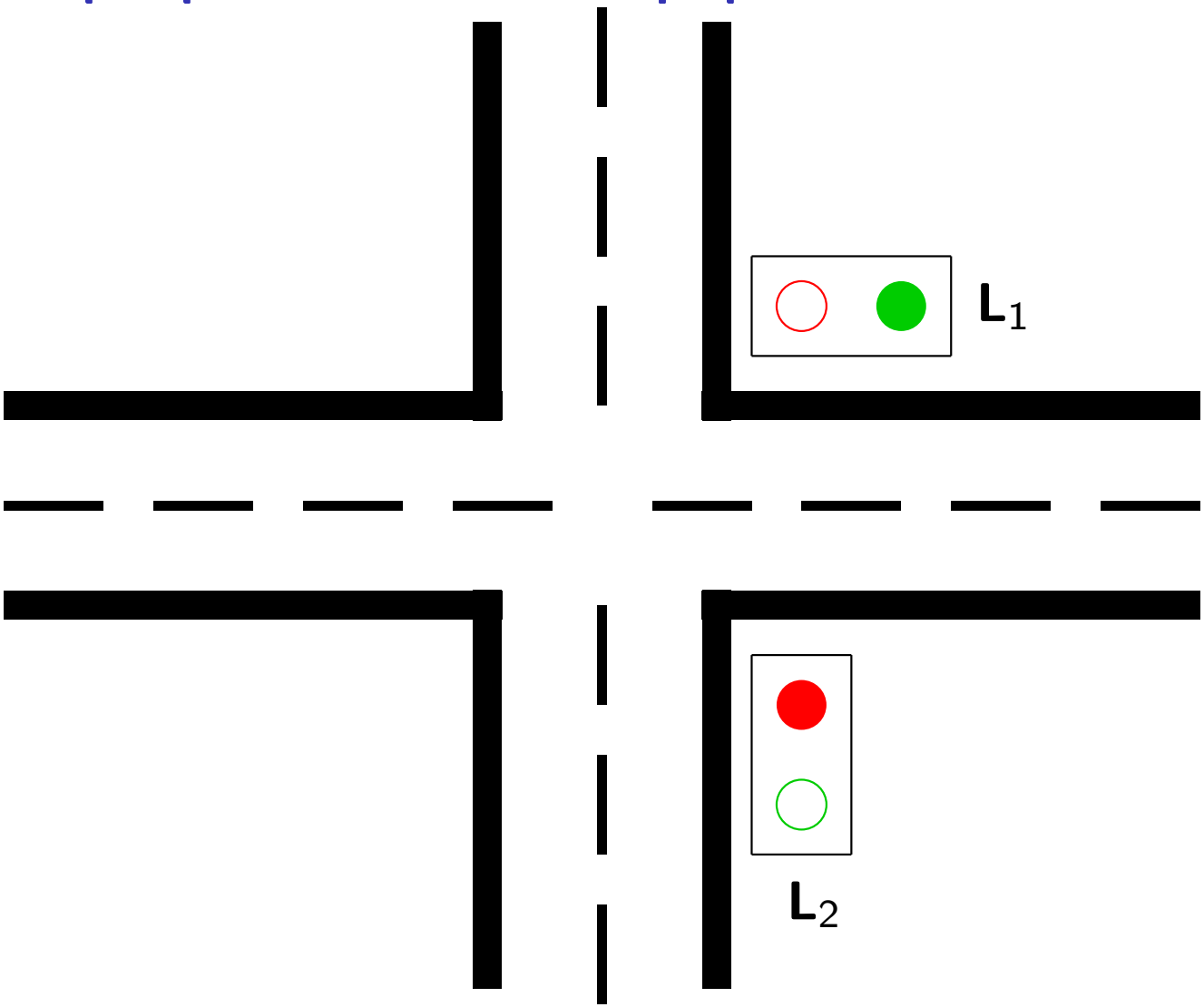
Свойства вычислений программ

Перекресток со светофорами



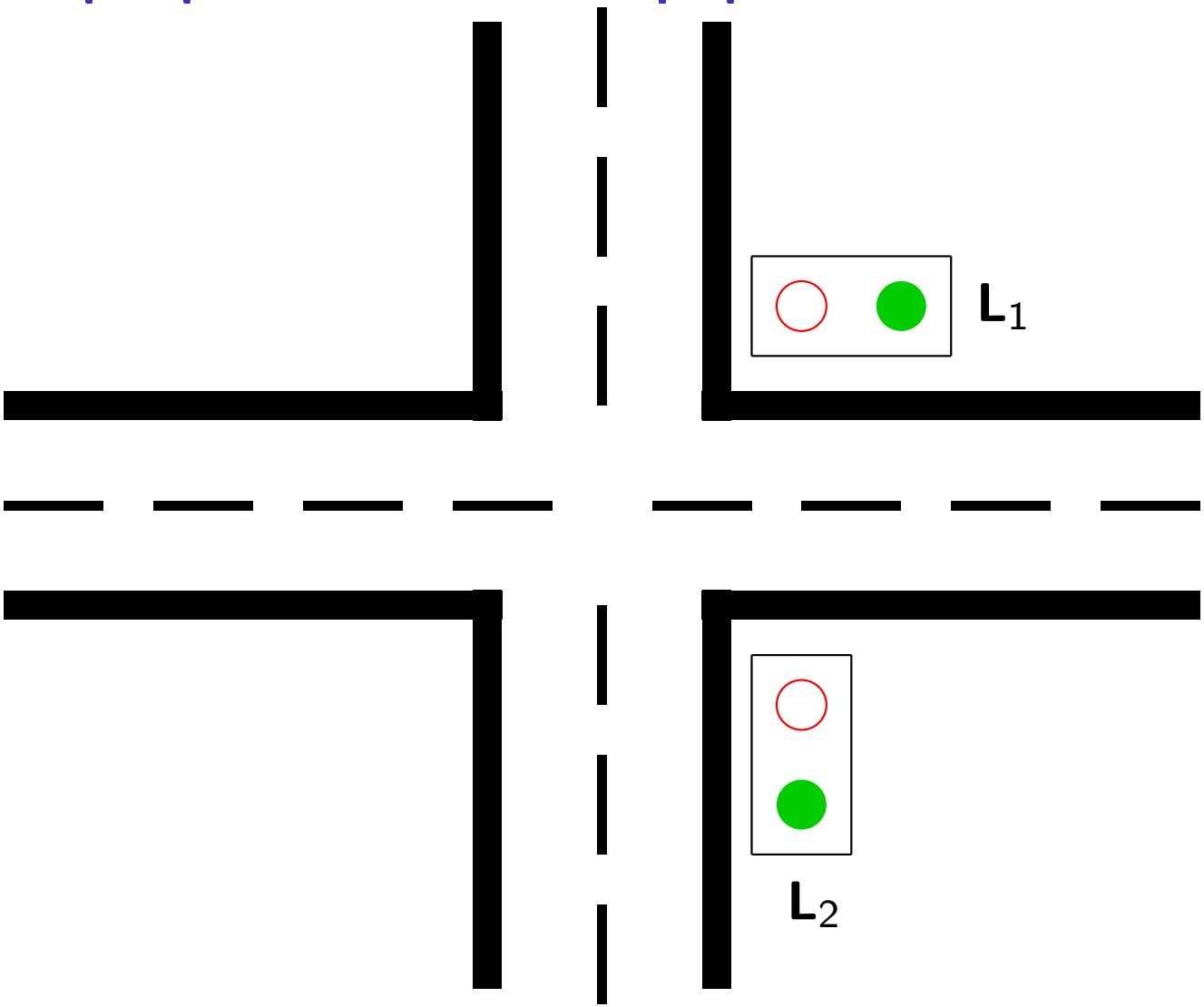
Свойства вычислений программ

Перекресток со светофорами



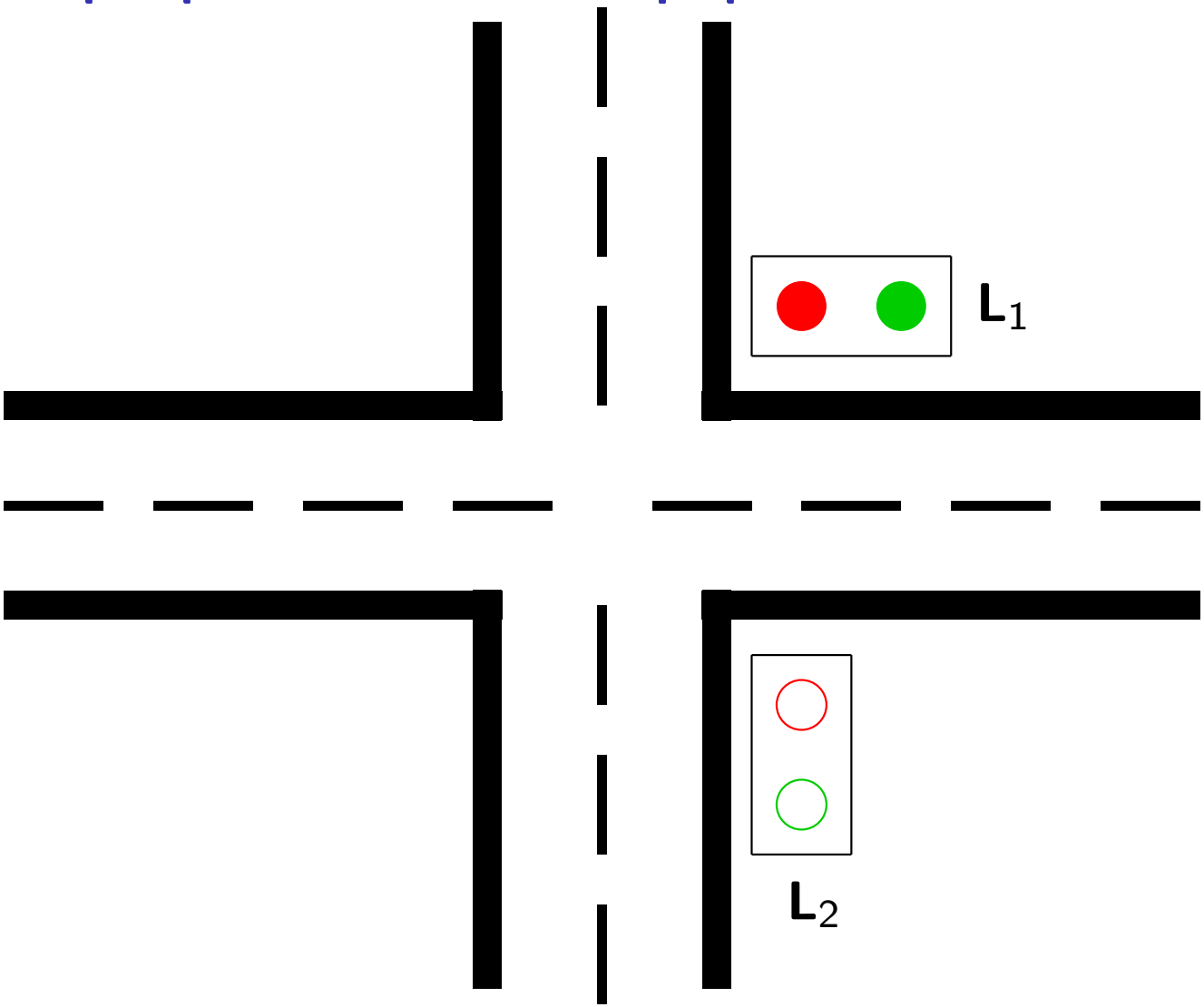
Свойства вычислений программ

Перекресток со светофорами



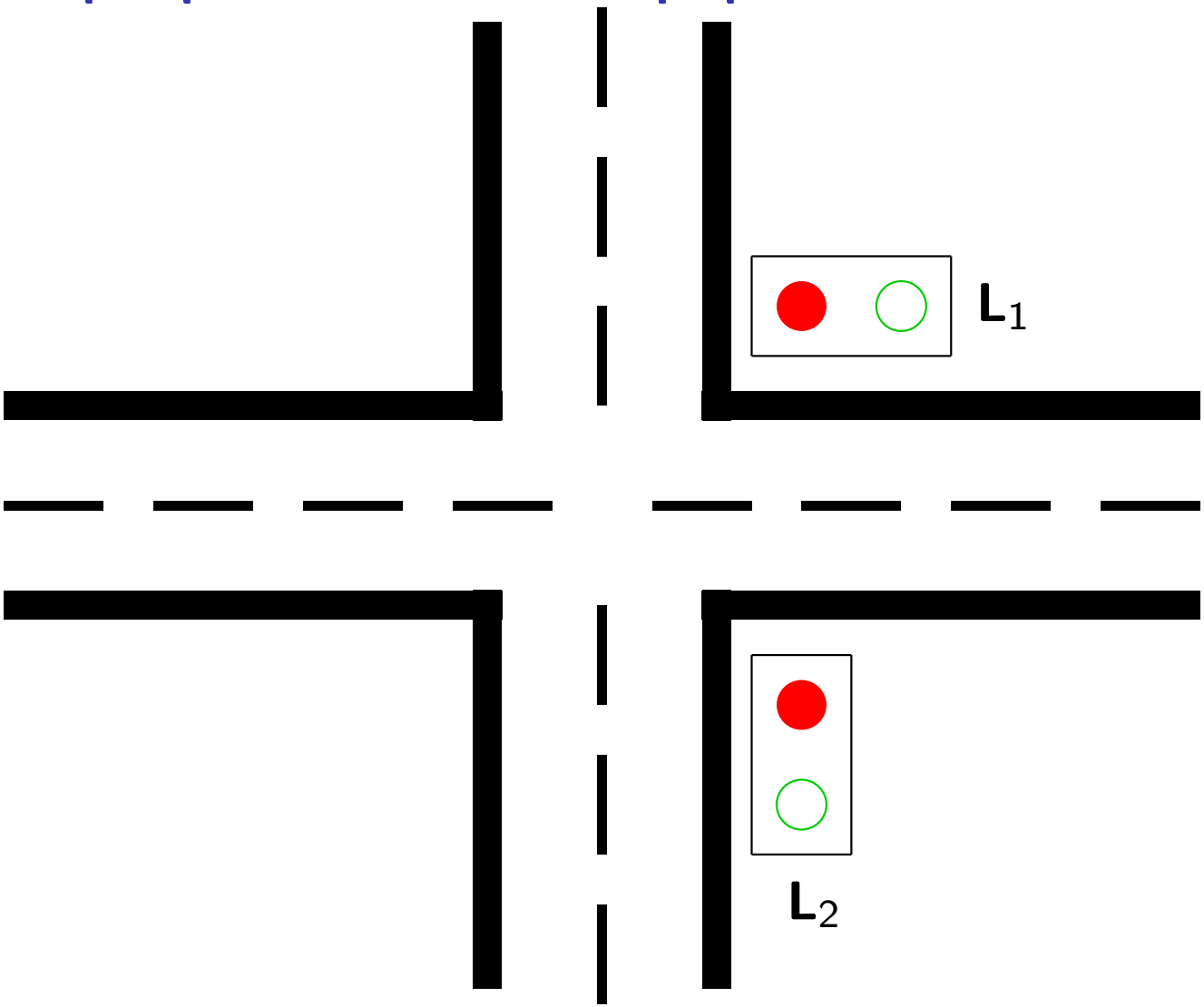
Свойства вычислений программ

Перекресток со светофорами



Свойства вычислений программ

Перекресток со светофорами



Свойства вычислений программ

Каждое состояние системы регулировки дорожного движения характеризуется событием — совокупностью активных цветов светофоров.

Свойства вычислений программ

Каждое состояние системы регулировки дорожного движения характеризуется событием — совокупностью активных цветов светофоров.

Каждое вычисление системы характеризуется трассой — последовательностью событий.

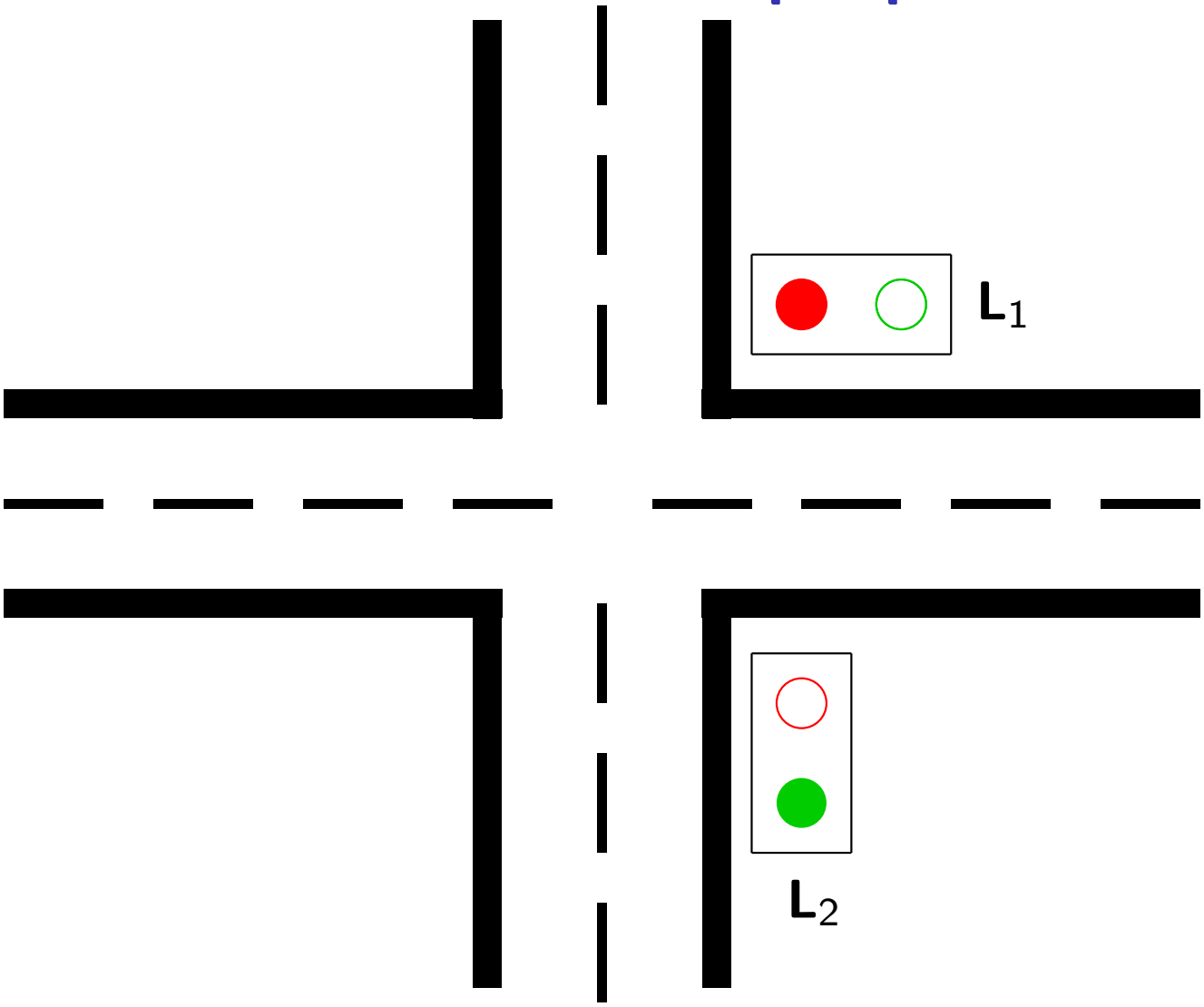
Свойства вычислений программ

Каждое состояние системы регулировки дорожного движения характеризуется событием — совокупностью активных цветов светофоров.

Каждое вычисление системы характеризуется трассой — последовательностью событий.

Поведение системы характеризуется свойством — множеством трасс.

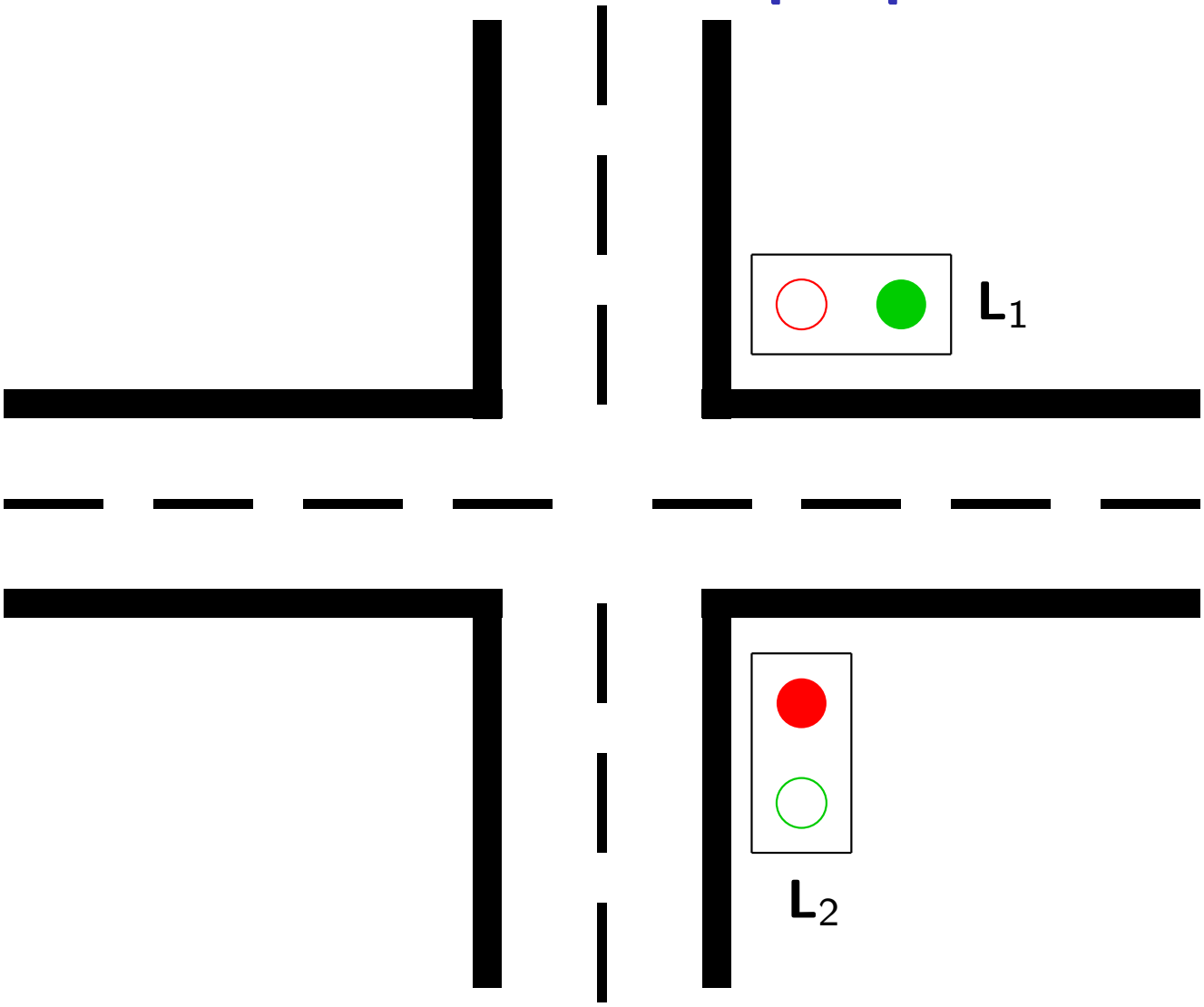
Свойства вычислений программ



Трасса:

$$\{red_1, green_2\},$$

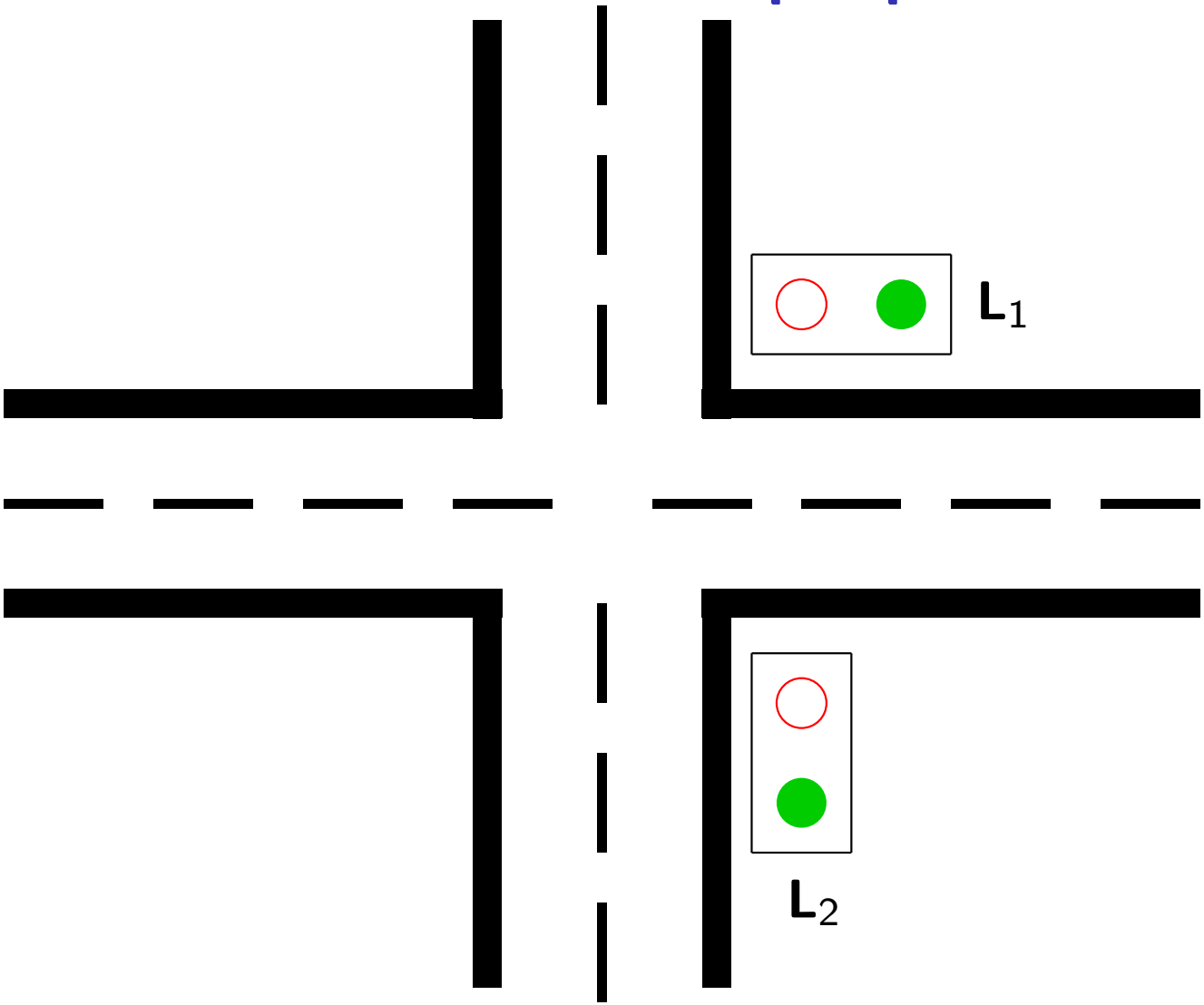
Свойства вычислений программ



Трасса:

$$\{red_1, green_2\}, \{green_1, red_2\},$$

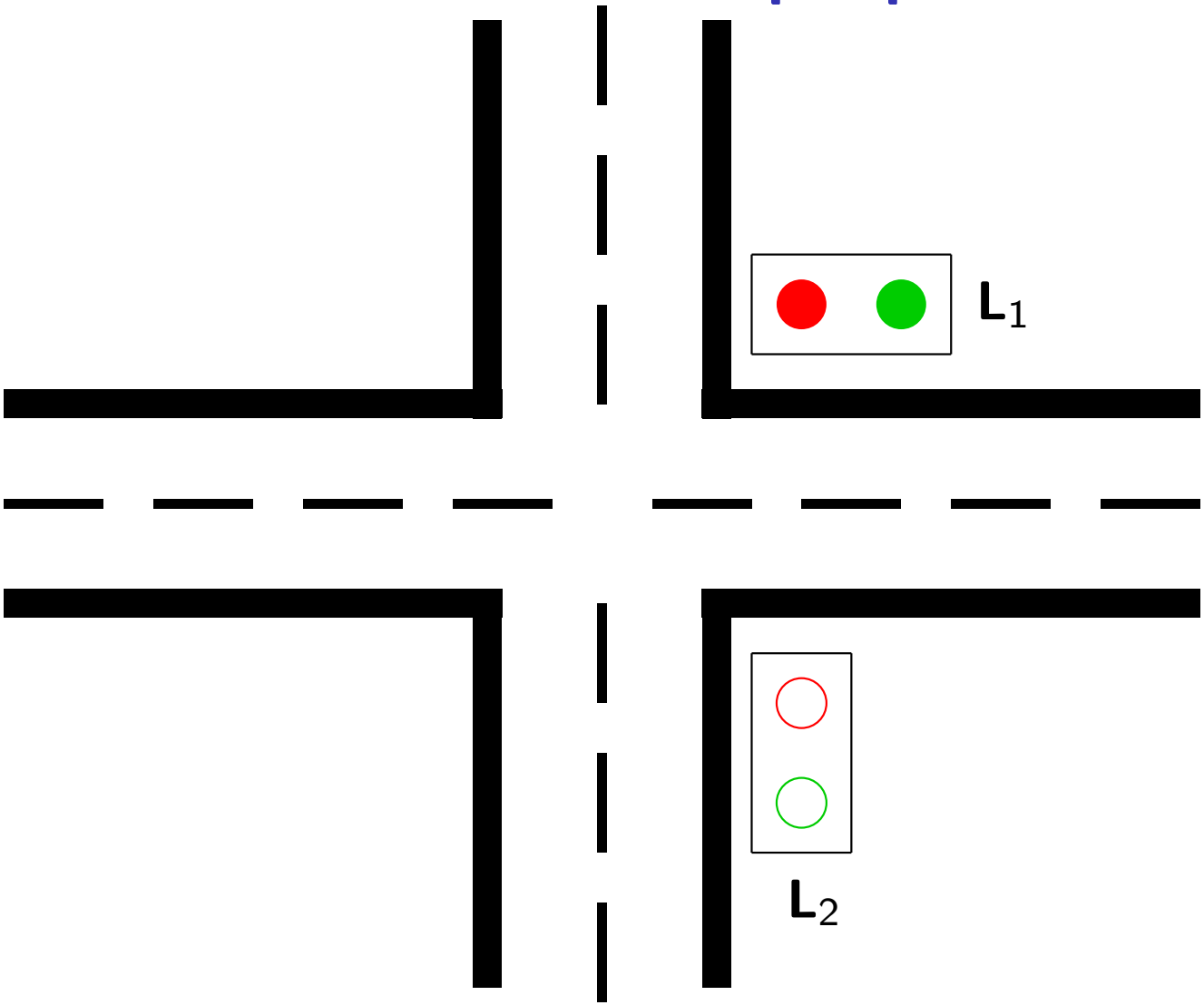
Свойства вычислений программ



Трасса:

$$\{red_1, green_2\}, \{green_1, red_2\}, \{green_1, green_2\}$$

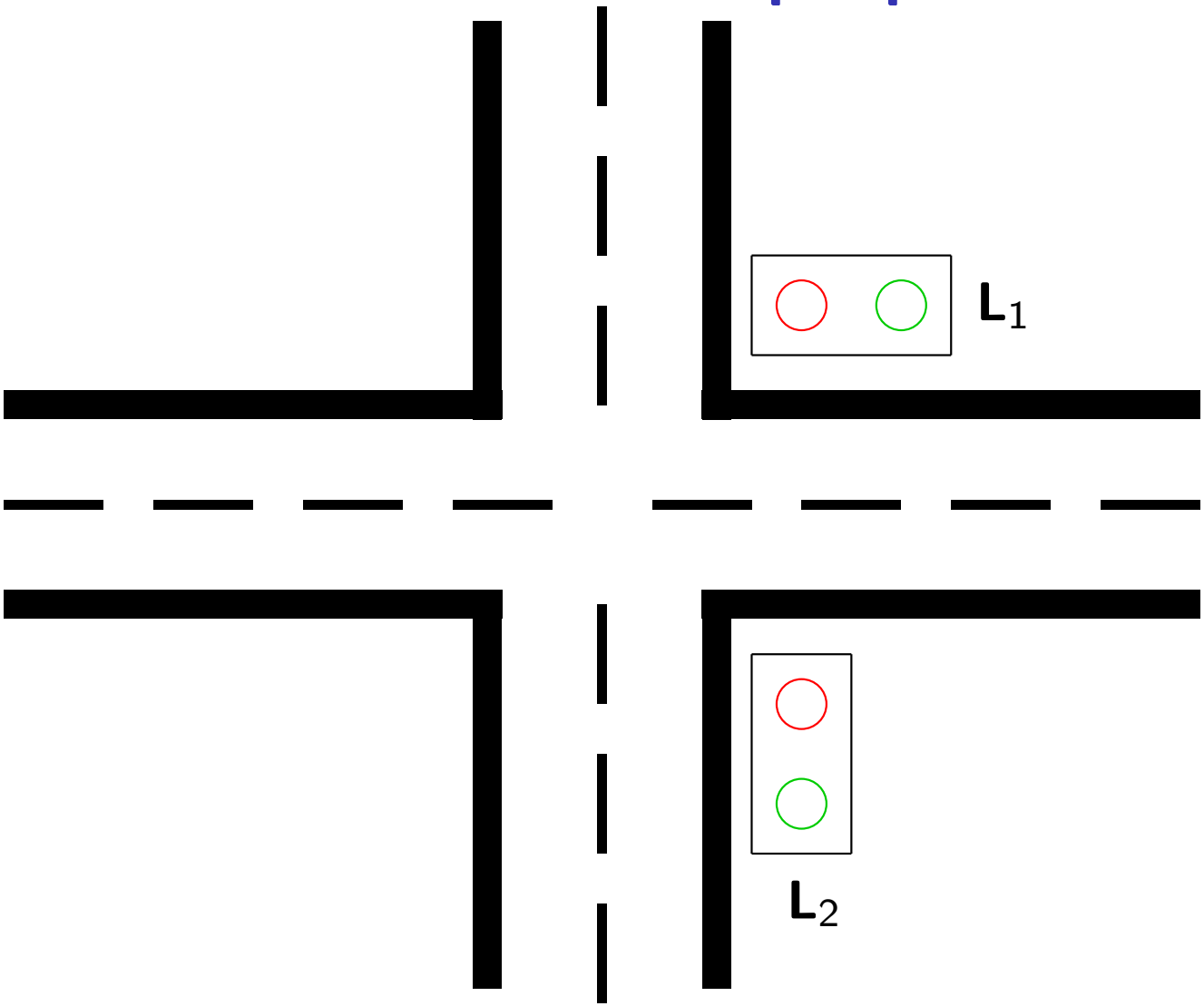
Свойства вычислений программ



Трасса:

$\{red_1, green_2\}, \{green_1, red_2\}, \{green_1, green_2\}\{red_1, green_1\},$

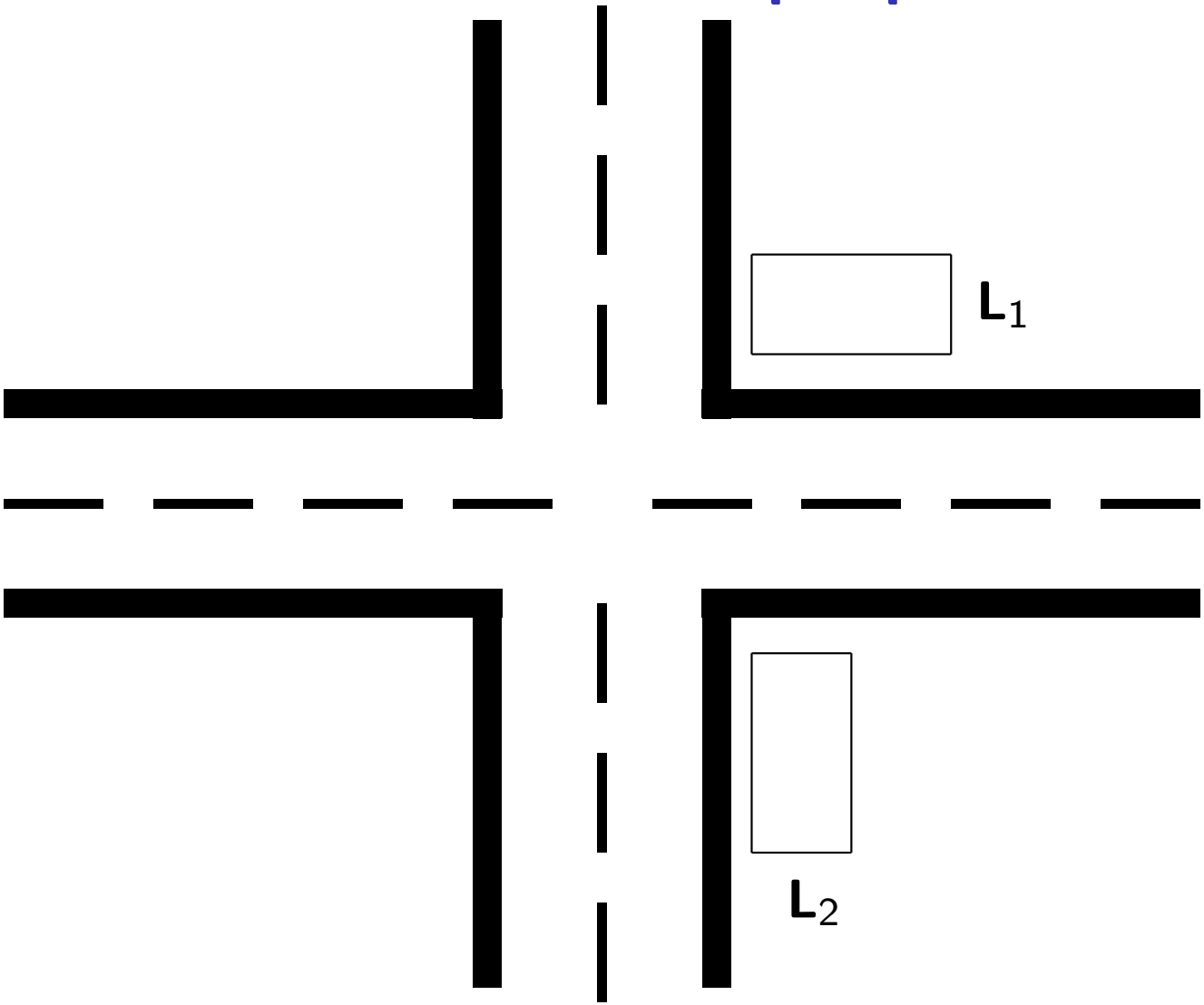
Свойства вычислений программ



Трасса:

$\{red_1, green_2\}, \{green_1, red_2\}, \{green_1, green_2\}\{red_1, green_1\}, ???$

Свойства вычислений программ



Трасса:

$\{red_1, green_2\}, \{green_1, red_2\}, \{green_1, green_2\}\{red_1, green_1\}, \emptyset,$

Свойства вычислений программ

Свойства системы регулировки дорожного движения:

Свойства вычислений программ

Свойства системы регулировки дорожного движения:

- У каждого светофора всегда активен в точности один цвет;

Свойства вычислений программ

Свойства системы регулировки дорожного движения:

- У каждого светофора всегда активен в точности один цвет;
- У каждого светофора рано или поздно активизируется зеленый цвет;

Свойства вычислений программ

Свойства системы регулировки дорожного движения:

- У каждого светофора всегда активен в точности один цвет;
- У каждого светофора рано или поздно активизируется зеленый цвет;
- У обоих светофоров никогда не бывает одновременно активен зеленый цвет;

Свойства вычислений программ

Свойства системы регулировки дорожного движения:

- У каждого светофора всегда активен в точности один цвет;
- У каждого светофора рано или поздно активизируется зеленый цвет;
- У обоих светофоров никогда не бывает одновременно активен зеленый цвет;
- У каждого светофора перемена активных цветов случается бесконечно часто.

Свойства вычислений программ

Более формально все эти понятия определяются так.

Пусть задано множество атомарных высказываний AP .

Тогда

- событие — любое подмножество атомарных высказываний, $E,\ E \subseteq AP$;
- трасса — любая бесконечная последовательность событий, $\alpha,\ \alpha \in (2^{AP})^{\omega}$,
$$\alpha = E_0, E_1, E_2, \ldots;$$
- свойство вычислений — любое множество трасс, $P,\ P \subseteq (2^{AP})^{\omega}$.

Свойства вычислений программ

Пусть задана модель Крипке $M = (S, S_0, R, L)$ и некоторый путь $\pi = s_0, s_1, s_2, \ldots$ из начального состояния s_0, $s_0 \in S_0$ в этой модели. Тогда трассой пути π назовем последовательность

$$\alpha(\pi) = L(s_0), L(s_1), L(s_2), \ldots$$

множеств атомарных высказываний, истинных в состояниях этого пути.

Совокупность всех трасс модели Крипке M обозначим записью $Tr(M)$.

Свойства вычислений программ

Пусть задана модель Крипке $M = (S, S_0, R, L)$ и некоторый путь $\pi = s_0, s_1, s_2, \ldots$ из начального состояния s_0, $s_0 \in S_0$ в этой модели. Тогда трассой пути π назовем последовательность

$$\alpha(\pi) = L(s_0), L(s_1), L(s_2), \ldots$$

множеств атомарных высказываний, истинных в состояниях этого пути.

Совокупность всех трасс модели Крипке M обозначим записью $Tr(M)$.

Говорят, что модель M удовлетворяет свойству P (обозначается $M \models P$), если верно включение $Tr(M) \subseteq P$.

Свойства вычислений программ

Модель Крипке M_1 называется уточнением модели M_2, если $Tr(M_1) \subseteq Tr(M_2)$.

Утверждение 1.

Если модель Крипке удовлетворяет некоторому свойству, то и всякое ее уточнение также удовлетворяет этому свойству.

Доказательство.

Самостоятельно.

Классификация свойств вычислений

Наиболее распространенные свойства вычислений программ разделяются на следующие классы:

- свойства безопасности (**safety**),
- свойства живости (**liveness**),
- ограничения справедливости (**fairness constraints**),

Классификация свойств вычислений

Свойство безопасности: «ничего плохого никогда не случится».

Свойство вычислений P называется свойством безопасности , если оно удовлетворяет следующему требованию:

- какова бы ни была трасса α, $\alpha \in (2^{AP})^{\omega} \setminus P$, существует такой ее конечный префикс β , что для любой трассы α' выполняется соотношение

$$\beta\alpha' \notin P.$$

Классификация свойств вычислений

Свойство безопасности: «ничего плохого никогда не случится».

Свойство вычислений P называется свойством безопасности , если оно удовлетворяет следующему требованию:

- какова бы ни была трасса α, $\alpha \in (2^{AP})^{\omega} \setminus P$, существует такой ее конечный префикс β , что для любой трассы α' выполняется соотношение

$$\beta\alpha' \notin P.$$

Содержательный смысл: «отсутствие безопасности — это когда если что-то плохое случилось, то этого уже не исправить».

Классификация свойств вычислений

Примеры свойств безопасности:

- В светофоре красный свет загорается только после желтого.
- Пока принтер не завершит печать, он не доступен для других устройств.
- Два процесса никогда не могут обращаться одновременно к одной и той же области памяти.
- Протокол раздвижного окна никогда не теряет первую порцию данных при передаче.

Классификация свойств вычислений

Свойство живости: «что-то хорошее обязательно произойдет».

Свойство вычислений P называется свойством живости , если оно удовлетворяет следующему требованию:

- для любого конечного слова $\beta,\ \beta \in (2^{AP})^*$, существует такая трасса $\alpha,\ \alpha \in (2^{AP})^\omega$, для которой выполняется соотношение

$$\beta\alpha \in P.$$

Классификация свойств вычислений

Свойство живости: «что-то хорошее обязательно произойдет».

Свойство вычислений P называется свойством живости , если оно удовлетворяет следующему требованию:

- для любого конечного слова $\beta,\ \beta \in (2^{AP})^*$, существует такая трасса $\alpha,\ \alpha \in (2^{AP})^\omega$, для которой выполняется соотношение

$$\beta\alpha \in P.$$

Содержательный смысл: «что бы ни случилось вначале, потом всегда можно достичь своей цели».

Классификация свойств вычислений

Примеры свойств живости:

- В светофоре когда-нибудь загорится зеленый свет.
- После того как принтер завершает печать, он стирает содержимое своего буфера.
- Процесс может бесконечно часто обращаться к заданной области памяти.
- Протокол раздвижного окна обязательно доставляет первую порцию данных по назначению.

Классификация свойств вычислений

Утверждение 2.

Если свойство P является как свойством живости, так и свойством безопасности, то $P = (2^{AP})^{\omega}$.

Доказательство.

Самостоятельно .

Классификация свойств вычислений

Утверждение 2.

Если свойство P является как свойством живости, так и свойством безопасности, то $P = (2^{AP})^{\omega}$.

Доказательство.

Самостоятельно .

Утверждение 3.

Для любого свойства P существуют такие свойства живости P_{live} и безопасности P_{safe} , для которых верно равенство

$$P = P_{live} \cap P_{safe}$$

Доказательство.

Самостоятельно [Трудная задача! Будет высоко оценена!] .

Классификация свойств вычислений

А к какому классу относится такое свойство вычислений:

«После того, как вначале процесс откроет файл, далее он может бесконечно часто проводить из него считывание данных»?

Классификация свойств вычислений

А к какому классу относится такое свойство вычислений:

«После того, как вначале процесс откроет файл, далее он может бесконечно часто проводить из него считывание данных»?

Оно не является ни свойством безопасности, ни свойством живости.

Ограничения справедливости

При моделировании вычислений информационных систем, состоящих из нескольких параллельно работающих процессов, применяется принцип чередования (interleaving assumption):

параллельная композиция вычислений

$$\begin{aligned} comp_1 &= a_1, a_2, a_3, \ldots \\ comp_2 &= b_1, b_2, b_3, \ldots \end{aligned}$$

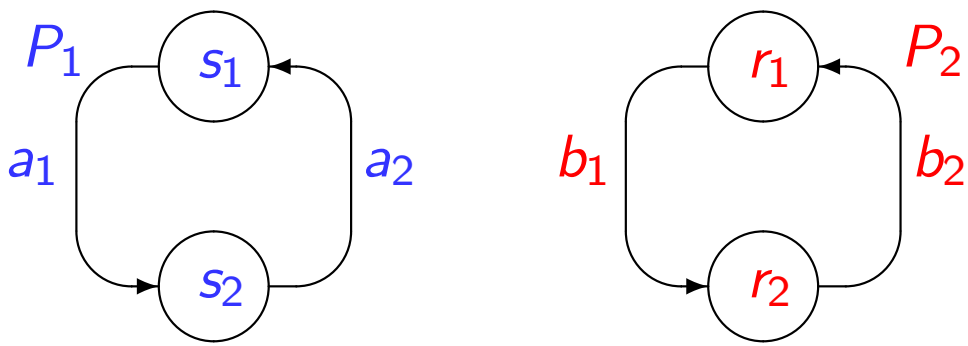
представляется множеством **чередований** последовательностей $comp_1$ и $comp_2$ — перемешиваний, сохраняющих относительный порядок следования членов одной и той же последовательности.

Пример чередования: $a_1, b_1, b_2, a_2, b_3, a_3, a_4, \ldots$.

Ограничения справедливости

Однако при построении параллельной композиции моделей процессов принцип чередования соблюдается не в полной мере.

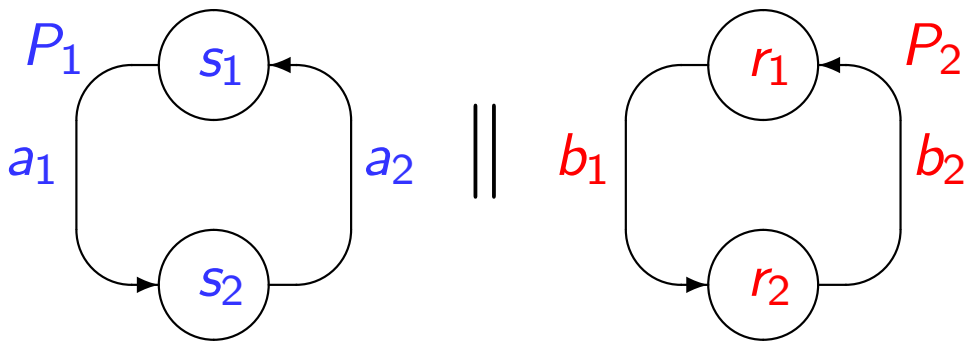
Рассмотрим пару моделей Крипке для процессов P_1 и P_2.



Ограничения справедливости

Однако при построении параллельной композиции моделей процессов принцип чередования соблюдается не в полной мере.

Рассмотрим пару моделей Крипке для процессов P_1 и P_2.



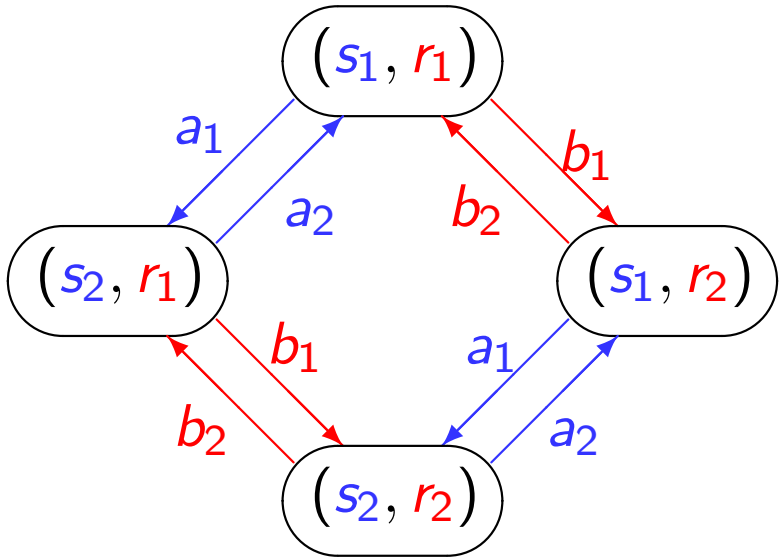
Параллельная асинхронная композиция этих процессов не может иметь вычисления

$$comp(P_1 \parallel P_2) = a_1, a_2, a_1, a_2, \ldots \textbf{ и.т.д.}$$

поскольку оно не является результатом чередования каких-либо двух вычислений этих процессов.

Ограничения справедливости

Но если построить модель Крипке для параллельной асинхронной композиции $P_1 \parallel P_2$,



то в этой модели имеется путь

$$\pi = a_1, a_2, a_1, a_2, \ldots \textbf{ и.т.д.}$$

Значит, нужно средство, позволяющее устранить недопустимые («нечестные») пути в моделях Крипке.

Ограничения справедливости

Ограничения справедливости — это условия, которым должны удовлетворять пути в моделях Крипке, чтобы эти пути соответствовали вычислениям, построенным по принципу чередования.

Каждое ограничение справедливости относится к некоторому действию (переходу) в модели и требует обязательного выполнения этого действия в вычислении модели в зависимости от того, насколько часто выполняются условия для совершения этого действия.

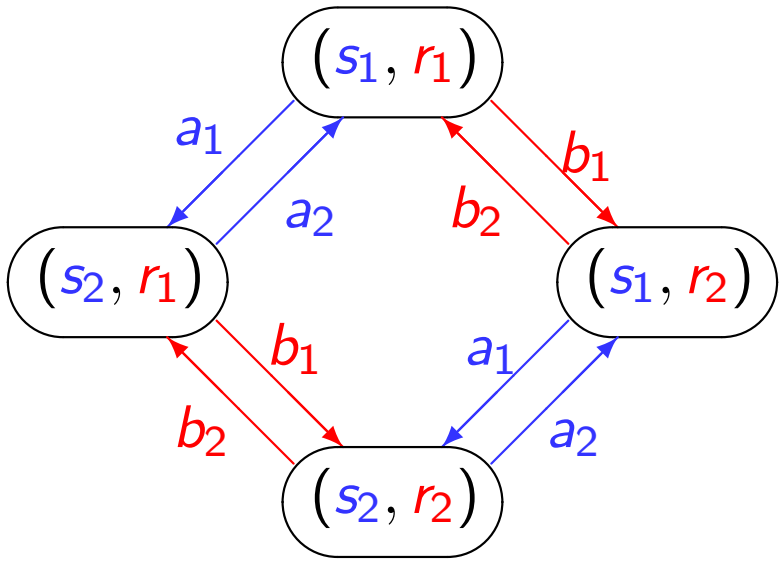
Ограничения справедливости

Различают два типа ограничений справедливости.

- Слабая справедливость : если путь **почти всегда** проходит через состояния, в которых может быть выполнено действие *act* , то действие *act* должно быть выполнено **бесконечно часто**.
- Сильная справедливость : если путь **бесконечно часто** проходит через состояния, в которых может быть выполнено действие *act* , то действие *act* должно быть выполнено **бесконечно часто**.

Ограничения справедливости

Таким образом, в модели Крипке для параллельной асинхронной композиции $P_1 \parallel P_2$



путь

$$\pi = a_1, a_2, a_1, a_2, \ldots \textbf{ и.т.д.}$$

не удовлетворяет ограничению слабой справедливости относительно действия b_1 , и поэтому не моделирует никакое вычисление.

Ограничения справедливости

Справедливая модель Крипке — это система размеченных переходов $M = (S, S_0, R, L, Act)$ с ограничениями справедливости $Fair = (WeakFair, StrongFair)$, где

- Act — множество действий
- R — отношение переходов, $R \subseteq S \times Act \times S$
- WeakFair — множество действий, подпадающих под ограничение слабой справедливости, $WeakFair \subseteq Act$;
- StrongFair — множество действий, подпадающих под ограничение слабой справедливости, $StrongFair \subseteq Act$.

Ограничения справедливости

Справедливая модель Крипке — это система размеченных переходов $M = (S, S_0, R, L, Act)$ с ограничениями справедливости $Fair = (WeakFair, StrongFair)$, где

- Act — множество действий
- R — отношение переходов, $R \subseteq S \times Act \times S$
- WeakFair — множество действий, подпадающих под ограничение слабой справедливости, $WeakFair \subseteq Act$;
- StrongFair — множество действий, подпадающих под ограничение слабой справедливости, $StrongFair \subseteq Act$.

Обозначим записью $Tr(M, Fair)$ множество трасс всех тех инициальных путей в модели M , которые удовлетворяют ограничениям справедливости $Fair$.

Тогда задача верификации моделей программ состоит в проверке соотношений вида

$$Tr(M, Fair) \subseteq P,$$

которые также обозначаются записью $M, Fair \models P$.

Темпоральные логики

Темпоральные логики предназначены для описания свойств вычислений реагирующих систем, т.е. множеств трасс в размеченных системах переходов (моделях Крипке).

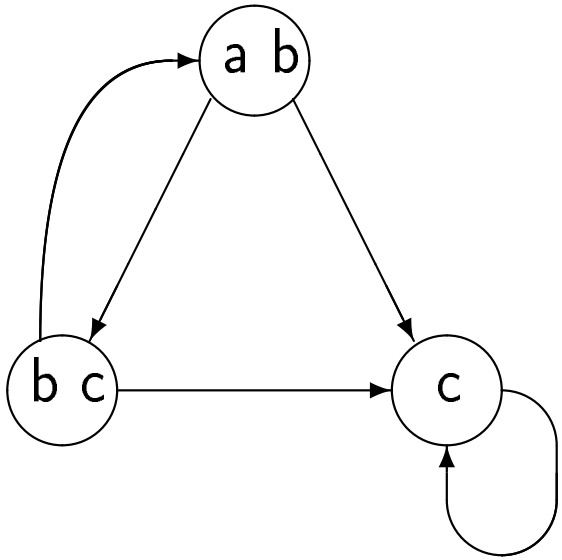
В темпоральных логиках время явно не упоминается; вместо этого формулы позволяют, например, записывать утверждения о том, что некоторое выделенное состояние будет когда-нибудь пройдено или что состояние ошибки никогда не будет достигнуто.

Для этого используются специальные темпоральные операторы. Темпоральные логики отличаются набором используемых темпоральных операторов и семантикой этих операторов.

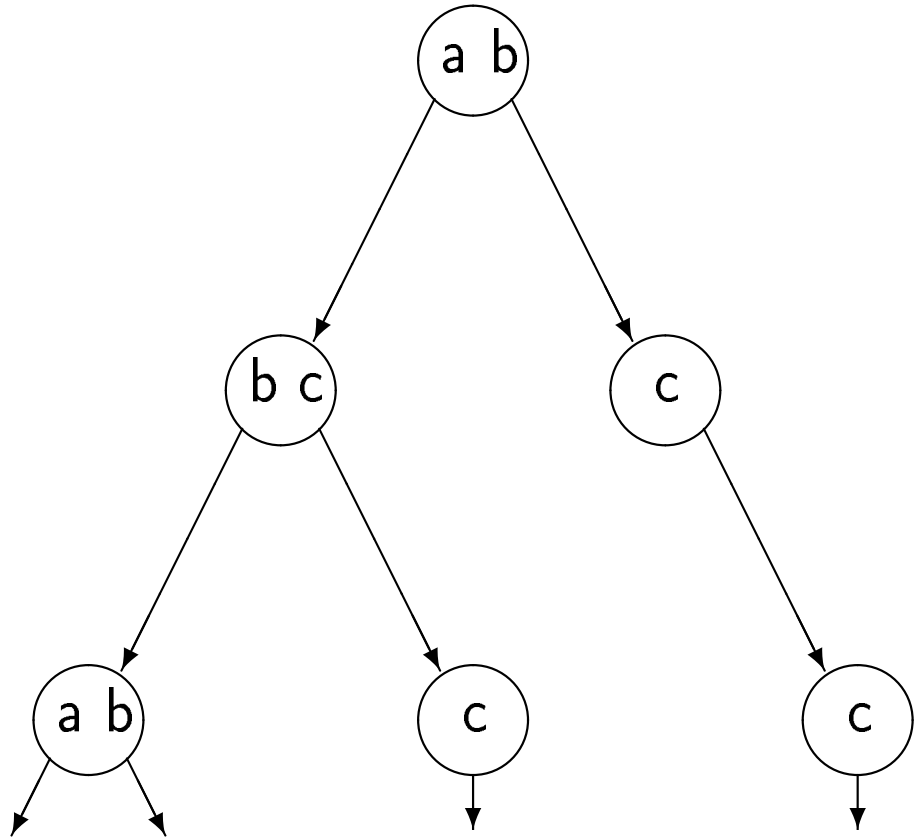
В центре нашего внимания будет очень выразительная логика под названием CTL*.

Логика деревьев вычислений CTL*

Формулы CTL* описывают свойства деревьев вычислений. Такое дерево образуется за счет выделения некоторого состояния в модели Крипке в качестве начального состояния и последующей развертки модели в бесконечное дерево с корнем в выделенном состоянии.



Граф переходов, или модель Крипке



Бесконечное дерево, развернутое из графа переходов

Логика деревьев вычислений CTL*

Формулы CTL* строятся из кванторов пути и темпоральных операторов.

Кванторы пути **A** («для всех путей вычисления») и **E** («для некоторого пути вычисления») применяются для описания структуры ветвления в дереве вычислений. Эти кванторы используются в том или ином состоянии для указания того, что все пути или некоторые из путей, исходящих из данного состояния, обладают предписанным свойством.

Темпоральные операторы описывают свойства пути, проходящего в дереве.

Логика деревьев вычислений CTL*

Имеются три одноместных оператора.

- Оператор сдвига по времени **X** (neXttime, «в следующий момент»): **X** P означает, что свойство P должно соблюдаться в следующем состоянии пути.
- Оператор происшествия **F** (Future, «рано или поздно», «когда-то в будущем»): **F** P означает, что свойство P будет соблюдаться в каком-то последующем состоянии пути.
- Оператор инвариантности **G** (Globally, «всегда», «повсюду»): **G** P означает, что свойство P должно соблюдается в каждом состоянии пути.

Логика деревьев вычислений CTL*

Имеются два двухместных операторов.

- Оператор условного ожидания **U** (Until, «до тех пор пока»): P_1 **U** P_2 означает, что на пути имеется состояние, в котором соблюдается второе свойство P_2 , и при этом каждое предшествующее на этом пути состояние обладает первым свойством P_1 .
- Оператор разблокировки **R** (Release, «высвободить»): P_1 **R** P_2 означает, что второе свойство P_2 должно выполняться на пути вплоть до такого состояния, в котором соблюдается первое свойство P_1 . Однако первое свойство P_1 не обязательно должно когда-нибудь быть выполнено.

Логика деревьев вычислений CTL*

В CTL* имеются формулы двух типов: формулы состояния (способные обращаться в истину в некотором состоянии) и формулы пути (способные быть истинными на протяжении некоторого пути).

Логика деревьев вычислений CTL*

В CTL* имеются формулы двух типов: формулы состояния (способные обращаться в истину в некотором состоянии) и формулы пути (способные быть истинными на протяжении некоторого пути).

Формулы состояния определяются следующими правилами:

- Если $p \in AP$, то p — формула состояния.
- Если f_1 и f_2 — формулы состояния, то $\neg f_1$, $f_1 \wedge f_2$ и $f_1 \vee f_2$ — формулы состояния.
- Если f — формула пути, то $\mathbf{E}\, f$ и $\mathbf{A}\, f$ — формулы состояния.

Логика деревьев вычислений CTL*

В CTL* имеются формулы двух типов: формулы состояния (способные обращаться в истину в некотором состоянии) и формулы пути (способные быть истинными на протяжении некоторого пути).

Формулы состояния определяются следующими правилами:

- Если $p \in AP$, то p — формула состояния.
- Если f_1 и f_2 — формулы состояния, то $\neg f_1$, $f_1 \wedge f_2$ и $f_1 \vee f_2$ — формулы состояния.
- Если f — формула пути, то $\mathbf{E}\, f$ и $\mathbf{A}\, f$ — формулы состояния.

Формулы пути определяются следующими правилами:

- Если f — формула состояния, то f — формула пути.
- Если g_1 и g_2 — формулы пути, то $\neg g_1$, $g_1 \wedge g_2$, $g_1 \vee g_2$, $\mathbf{X}\, g_1$, $\mathbf{F}\, g_1$, $\mathbf{G}\, g_1$, $g_1\, \mathbf{U}\, g_2$, $g_1\, \mathbf{R}\, g_2$ — формулы пути.

Логика деревьев вычислений CTL*

В CTL* имеются формулы двух типов: формулы состояния (способные обращаться в истину в некотором состоянии) и формулы пути (способные быть истинными на протяжении некоторого пути).

Формулы состояния определяются следующими правилами:

- Если $p \in AP$, то p — формула состояния.
- Если f_1 и f_2 — формулы состояния, то $\neg f_1$, $f_1 \wedge f_2$ и $f_1 \vee f_2$ — формулы состояния.
- Если f — формула пути, то $\mathbf{E}\, f$ и $\mathbf{A}\, f$ — формулы состояния.

Формулы пути определяются следующими правилами:

- Если f — формула состояния, то f — формула пути.
- Если g_1 и g_2 — формулы пути, то $\neg g_1$, $g_1 \wedge g_2$, $g_1 \vee g_2$, $\mathbf{X}\, g_1$, $\mathbf{F}\, g_1$, $\mathbf{G}\, g_1$, $g_1\, \mathbf{U}\, g_2$, $g_1\, \mathbf{R}\, g_2$ — формулы пути.

Формулами CTL* являются все формулы состояний, построенные согласно указанным правилам.

Логика деревьев вычислений CTL*

Семантика CTL* определется на моделях Крипке $M = (S, S_0, R, L)$. Путем в M назовем всякую такую бесконечную последовательность состояний $\pi = s_0, s_1, \ldots$, что $(s_i, s_{i+1}) \in R$ для каждого $i \geq 0$. Путь можно понимать как бесконечную ветвь в дереве вычислений, соответствующем модели M.

Мы будем использовать запись π^i для обозначения суффикса π, начинающегося состоянием s_i.

Если f — формула состояния, то запись $M, s \models f$ означает, что f выполняется в состоянии s на модели Крипке M. Точно так же, если g — формула пути, то запись $M, \pi \models g$ означает, что g выполняется на протяжении пути π в модели M.

Логика деревьев вычислений CTL*

Семантика формул состояния

1). $M, s \models p \Leftrightarrow p \in L(s)$;

2). $M, s \models \neg f_1 \Leftrightarrow M, s \not\models f_1$;

3). $M, s \models f_1 \vee f_2 \Leftrightarrow M, s \models f_1$ или $M, s \models f_2$;

4). $M, s \models f_1 \wedge f_2 \Leftrightarrow M, s \models f_1$ и $M, s \models f_2$;

5). $M, s \models \mathbf{E}\, f \Leftrightarrow$ в M есть такой путь из состояния s, что $M, \pi \models f$;

6). $M, s \models \mathbf{A}\, f \Leftrightarrow$ для любого пути π из состояния s в модели M верно соотношение $M, \pi \models f$;

Логика деревьев вычислений CTL*

Семантика формул пути

7). $M, \pi \models g_1 \Leftrightarrow$ для первого состояния s на пути π в модели M верно соотношение $M, s \models g_1$;

8). $M, \pi \models \neg g_1 \Leftrightarrow M, \pi \not\models g_1$;

9). $M, \pi \models g_1 \vee g_2 \Leftrightarrow M, \pi \models g_1$ или $M, \pi \models g_2$;

10). $M, \pi \models g_1 \wedge g_2 \Leftrightarrow M, \pi \models g_1$ и $M, \pi \models g_2$;

11). $M, \pi \models \mathbf{X}\, g_1 \Leftrightarrow M, \pi^1 \models g_1$;

12). $M, \pi \models \mathbf{F}\, g_1 \Leftrightarrow$ существует такое $k \geqslant 0$, что $M, \pi^k \models g_1$;

13). $M, \pi \models \mathbf{G}\, g_1 \Leftrightarrow$ для любого $k \geqslant 0$ верно соотношение $M, \pi^k \models g_1$;

14). $M, \pi \models g_1\, \mathbf{U}\, g_2 \Leftrightarrow$ существует такое $k \geqslant 0$, что $M, \pi^k \models g_2$ и для каждого $0 \leqslant j < k$ верно соотношение $M, \pi^j \models g_1$;

15). $M, \pi \models g_1\, \mathbf{R}\, g_2 \Leftrightarrow$ каково бы ни было $j \geqslant 0$, если для каждого $i < j$ верно соотношение $M, \pi^i \not\models g_1$, то $M, \pi^j \models g_2$.

Логика деревьев вычислений CTL*

Легко убедиться в том, что операторов $\vee$, $\neg$, $\mathbf{X}$, $\mathbf{U}$, $\mathbf{E}$ достаточно для того, чтобы с их помощью выразить любую формулу CTL*:

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$,
- $f \,\mathbf{R}\, g \equiv \neg(\neg f \,\mathbf{U}\, \neg g)$,
- $\mathbf{F}\, f \equiv \mathit{True}\ \mathbf{U}\, f$,
- $\mathbf{G}\, f \equiv \neg\, \mathbf{F}\, \neg f$,
- $\mathbf{A}(f) \equiv \neg\, \mathbf{E}(\neg f)$.

CTL и LTL

Чаще всего используют два полезных подмножества CTL*: одна из этих логик называется логикой ветвящегося времени , а другая — логикой линейного времени . Различие между ними проявляется в том, как они относятся к ветвлению в упомянутом дереве вычислений. В логике ветвящегося времени темпоральные операторы находятся непосредственно под действием кванторов по тем путям, которые исходят из заданного состояния. В логике линейного времени операторы предназначены для описания событий на протяжении единственного пути вычисления.

CTL и LTL

Логика деревьев вычислений (CTL) представляет собой фрагмент CTL*, в котором каждый темпоральный оператор **X**, **F**, **G**, **U** и **R** должен следовать непосредственно за квантором пути. Говоря более строго, CTL — это подмножество CTL*, в котором структура формул пути ограничена следующим правилом.

- Если f и g — формулы состояния, то $\mathbf{X}\, f$, $\mathbf{F}\, f$, $\mathbf{G}\, f$, $f\, \mathbf{U}\, g$, и $f\, \mathbf{R}\, g$ — формулы пути.

Таким образом, в формулах CTL каждый темпоральный оператор непосредственно следует за квантором пути.

Примеры.

$\mathbf{A}\, \mathbf{G}\, p_1 \rightarrow (p_2\ \mathbf{E}\, \mathbf{U}\, (\mathbf{A}\, \mathbf{F}\, p_3))$,

$\mathbf{E}\, \mathbf{F}\, p \vee \mathbf{A}\, \mathbf{G}(p \wedge q)$.

CTL и LTL

Линейная темпоральная логика (LTL), в свою очередь, состоит из всех формул вида $\mathbf{A}\, f$, где f — формула пути, в которой все формулы состояния — это атомарные высказывания. Более строгое определение формул пути LTL таково.

- Если $p \in AP$, то f — формула пути.
- Если f и g — формулы пути, то $\neg f$, $f \wedge g$, $f \vee g$, $\mathbf{X}\, f$, $\mathbf{F}\, f$, $\mathbf{G}\, f$, $f\, \mathbf{U}\, g$, $f\, \mathbf{R}\, g$ — формулы пути.

Примеры.

$\mathbf{A}(\mathbf{F}\, \mathbf{G}\, \mathit{enabled} \rightarrow \mathbf{G}\, \mathbf{F}\, \mathit{fired})$ — условие слабой справедливости;

$\mathbf{A}\, \mathbf{G}(\mathit{passive} \rightarrow (\mathit{active}\, \mathbf{U}\, \mathbf{X}\, \mathit{passive}))$.

CTL и LTL

Все три логики имеют разные выразительные возможности.

Утверждение 4.

Не существует CTL-формулы, эквивалентной LTL-формуле $\mathbf{A}\,(\mathbf{FG}\,\mathit{stable})$.

Эта формула выражает следующее свойство живости: при любом сценарии развития событий система рано или поздно стабилизируется и будет далее оставаться стабильной.

Доказательство.

Самостоятельно [Трудная задача! Будет высоко оценена!] .

CTL и LTL

Утверждение 5.

Не существует LTL формулы, которая была бы эквивалентна CTL-формуле **AG** (**EF** *restart*) .

Эта формула выражает следующее свойство: при любом сценарии развития событий система всегда имеет возможность перезагрузиться.

Доказательство.

Самостоятельно [Трудная задача! Будет высоко оценена!] .

CTL и LTL

Утверждение 5.

Не существует LTL формулы, которая была бы эквивалентна CTL-формуле $\mathbf{AG}\ (\mathbf{EF}\ \mathit{restart})$.

Эта формула выражает следующее свойство: при любом сценарии развития событий система всегда имеет возможность перезагрузиться.

Доказательство.

Самостоятельно [Трудная задача! Будет высоко оценена!] .

А дизъюнкция этих двух формул $\mathbf{A}\ (\mathbf{FG}\ p) \vee \mathbf{AG}\ (\mathbf{EF}\ p)$ является формулой CTL*, которую нельзя выразить ни в CTL, ни в LTL.

CTL

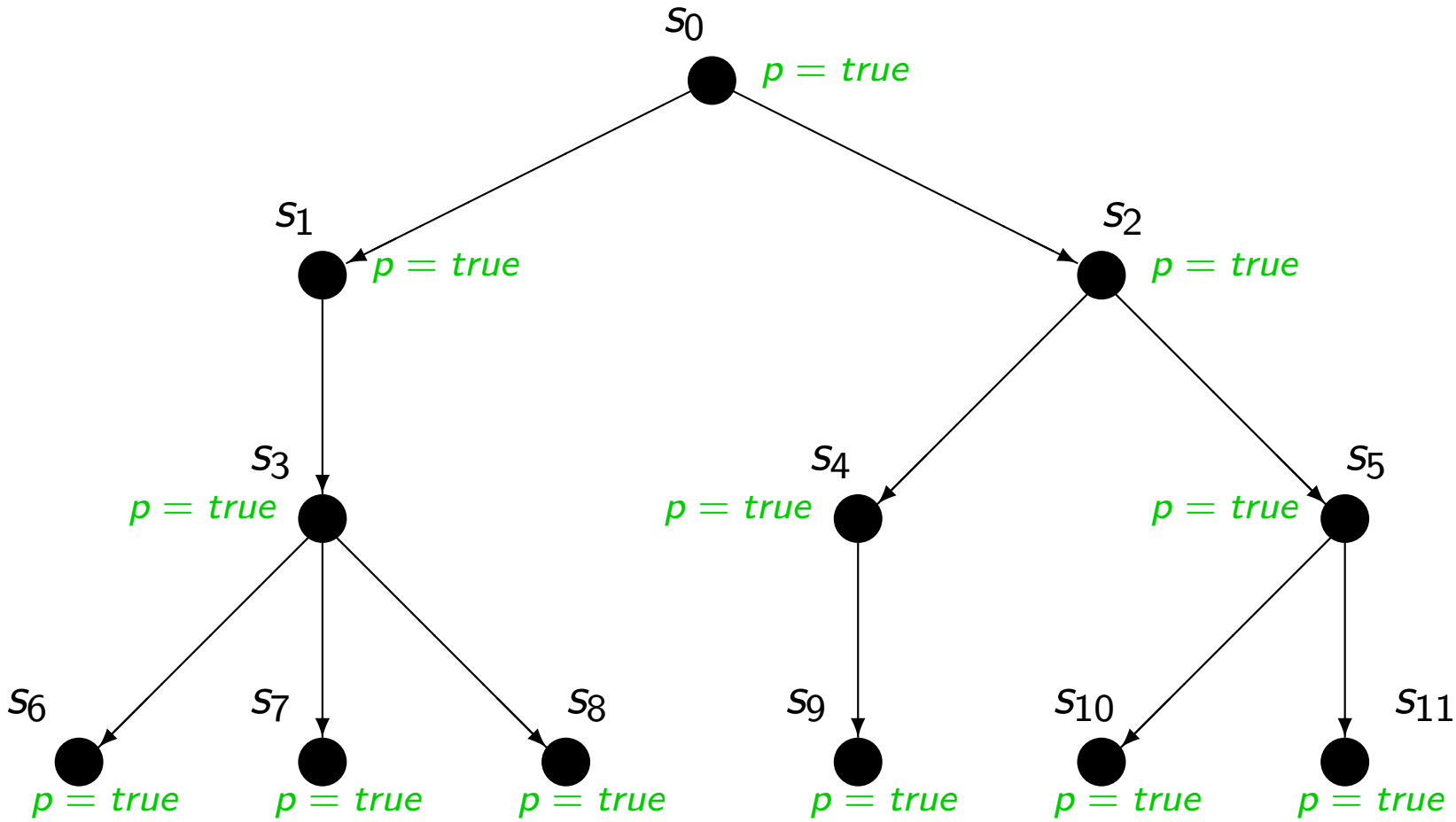
Мы ограничимся подробным изучением CTL. Имеются десять основных операторов CTL:

- **AX** и **EX**,
- **AF** и **EF**,
- **AG** и **EG**,
- **AU** и **EU**,
- **AR** и **ER**.

CTL

Логика деревьев вычислений CTL

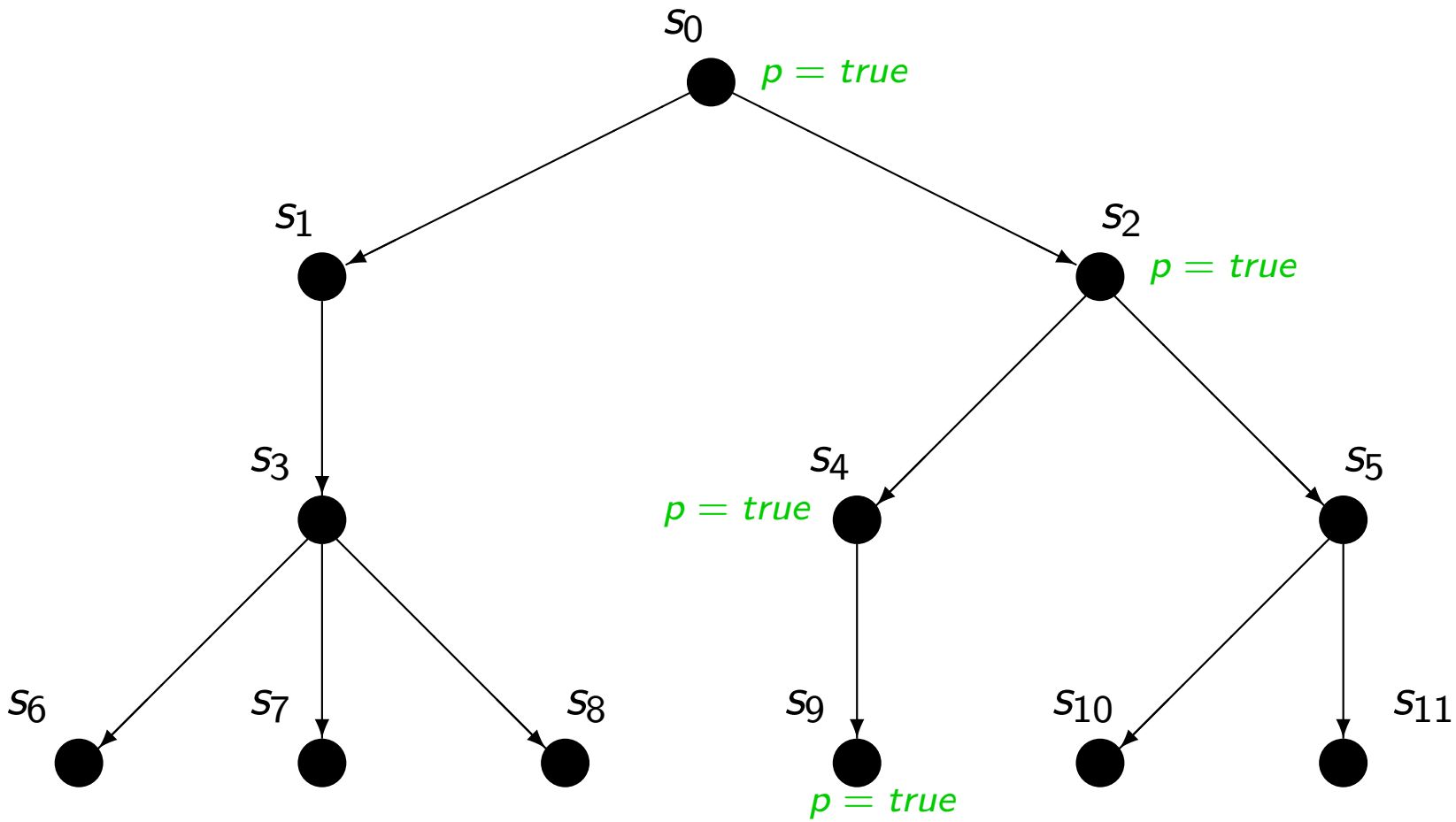
$$I, s_0 \models \mathbf{AG}p$$



CTL

Логика деревьев вычислений CTL

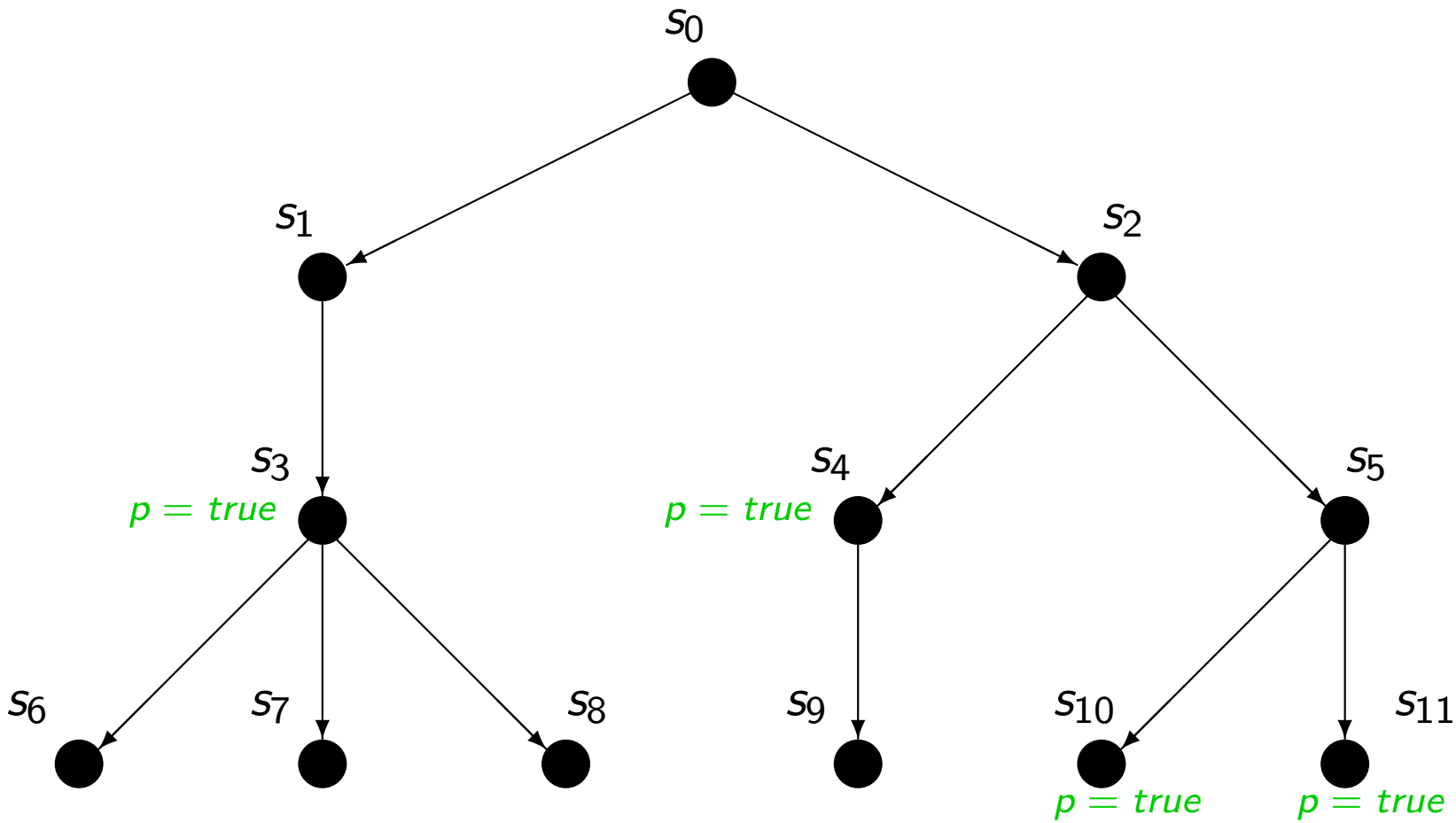
$$I, s_0 \models \mathbf{EG}p$$



CTL

Логика деревьев вычислений CTL

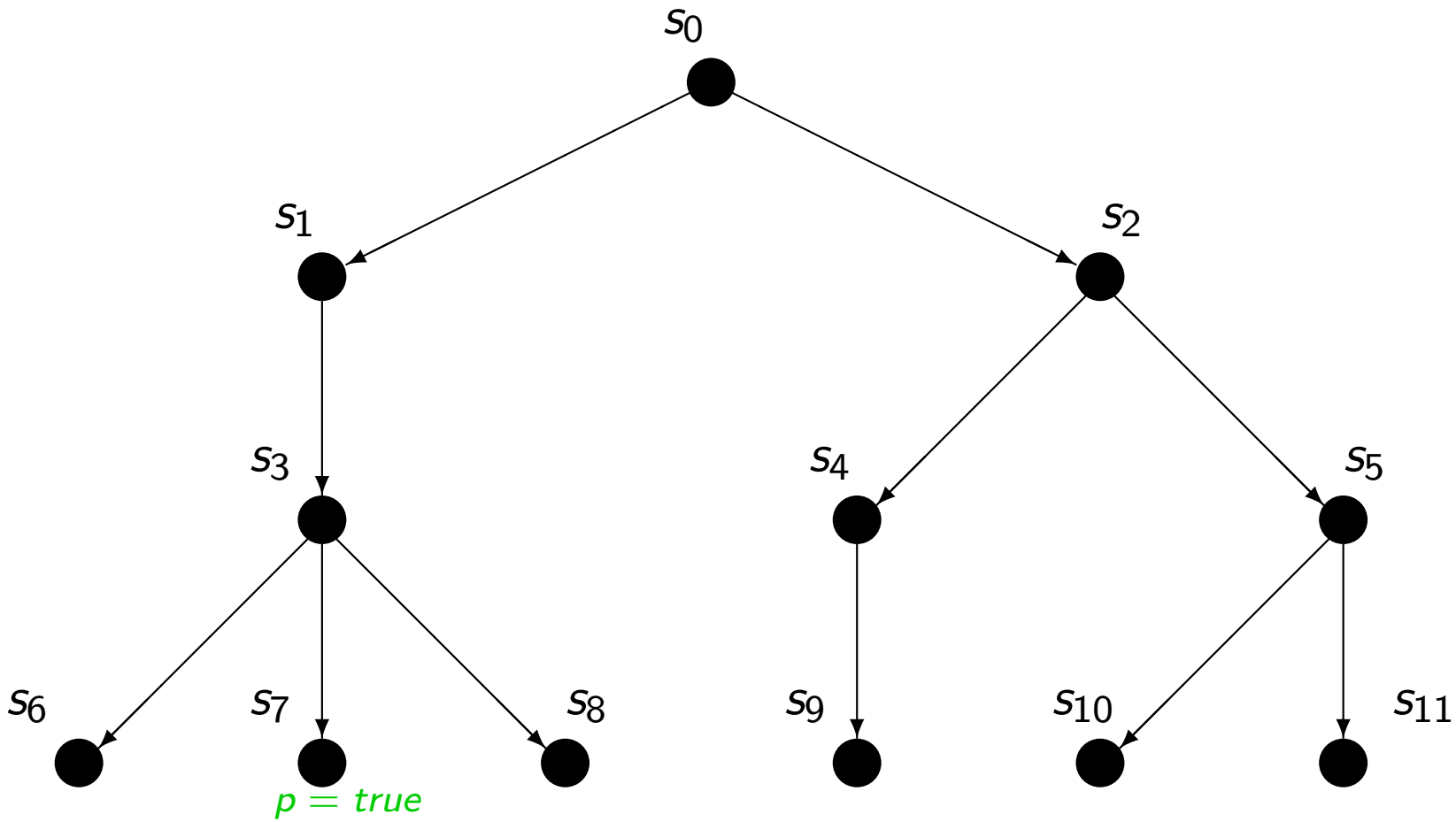
$$I, s_0 \models \mathbf{AF}p$$



CTL

Логика деревьев вычислений CTL

$$I, s_0 \models \mathbf{EF}p$$



Логика деревьев вычислений CTL

Каждый из этих операторов может быть выражен при помощи всего лишь трех операторов **EX**, **EG** и **EU**:

- $\mathbf{AX}\, f \equiv \neg\, \mathbf{EX}(\neg f)$;
- $\mathbf{EF}\, f \equiv \mathbf{E}[\mathit{True}\, \mathbf{U}\, f]$;
- $\mathbf{AG}\, f \equiv \neg\, \mathbf{EF}(\neg f)$;
- $\mathbf{AF}\, f \equiv \neg\, \mathbf{EG}(\neg f)$;
- $\mathbf{A}[f\, \mathbf{U}\, g] \equiv \neg\, \mathbf{E}[\neg g\, \mathbf{U}\, (\neg f \wedge \neg g)] \wedge \neg\, \mathbf{EG}\, \neg g$;
- $\mathbf{A}[f\, \mathbf{R}\, g] \equiv \neg\, \mathbf{E}[\neg f\, \mathbf{U}\, \neg g]$;
- $\mathbf{E}[f\, \mathbf{R}\, g] \equiv \neg\, \mathbf{A}[\neg f\, \mathbf{U}\, \neg g]$.

Логика деревьев вычислений CTL

Некоторые типичные CTL-формулы, которые возникают при верификации систем параллельных программ с конечным числом состояний, приводятся ниже:

Логика деревьев вычислений CTL

Некоторые типичные CTL-формулы, которые возникают при верификации систем параллельных программ с конечным числом состояний, приводятся ниже:

- **EF** $(Start \wedge \neg Ready)$: можно достичь такого состояния, в котором условие *Start* выполняется, а *Ready* — нет;

Логика деревьев вычислений CTL

Некоторые типичные CTL-формулы, которые возникают при верификации систем параллельных программ с конечным числом состояний, приводятся ниже:

- **EF** $(Start \wedge \neg Ready)$: можно достичь такого состояния, в котором условие *Start* выполняется, а *Ready* — нет;
- **AG** $(Req \rightarrow \mathbf{AF}\, Ack)$: когда бы ни был получен запрос, он рано или поздно будет подтвержден;

Логика деревьев вычислений CTL

Некоторые типичные CTL-формулы, которые возникают при верификации систем параллельных программ с конечным числом состояний, приводятся ниже:

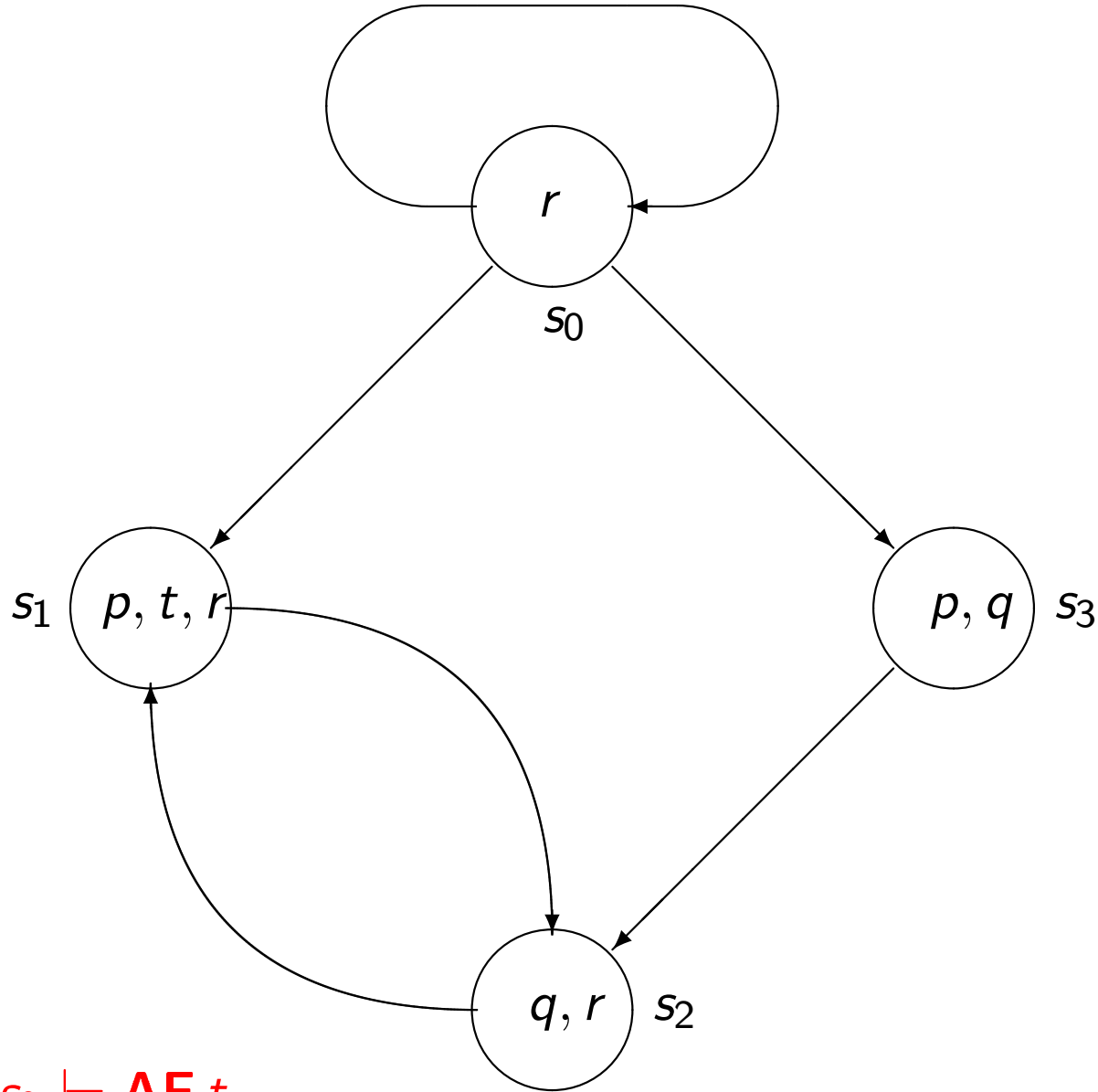
- **EF** $(Start \wedge \neg Ready)$: можно достичь такого состояния, в котором условие *Start* выполняется, а *Ready* — нет;
- **AG** $(Req \rightarrow \mathbf{AF}\, Ack)$: когда бы ни был получен запрос, он рано или поздно будет подтвержден;
- **AG** $(\mathbf{AF}\, DeviceEnabled)$: условие *DeviceEnabled* выполняется бесконечно часто на каждом пути вычисления;

Логика деревьев вычислений CTL

Некоторые типичные CTL-формулы, которые возникают при верификации систем параллельных программ с конечным числом состояний, приводятся ниже:

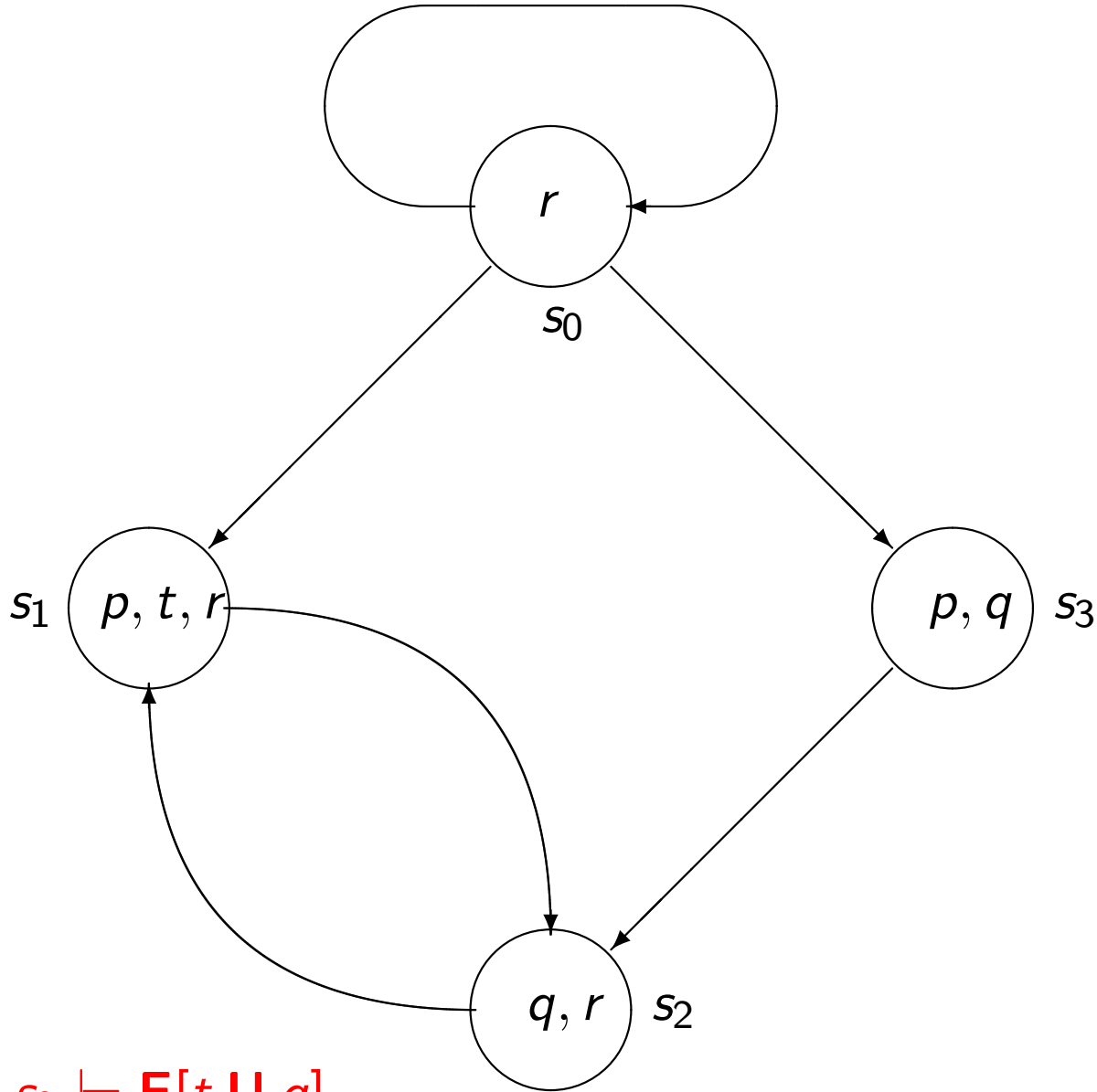
- **EF** $(Start \wedge \neg Ready)$: можно достичь такого состояния, в котором условие *Start* выполняется, а *Ready* — нет;
- **AG** $(Req \rightarrow \mathbf{AF}\, Ack)$: когда бы ни был получен запрос, он рано или поздно будет подтвержден;
- **AG** $(\mathbf{AF}\, DeviceEnabled)$: условие *DeviceEnabled* выполняется бесконечно часто на каждом пути вычисления;
- **AG** $(\mathbf{EF}\, Restart)$: из любого достожимого состояния достижимо состояние *Restart*.

Проверьте выполнимость формул CTL в заданной модели.



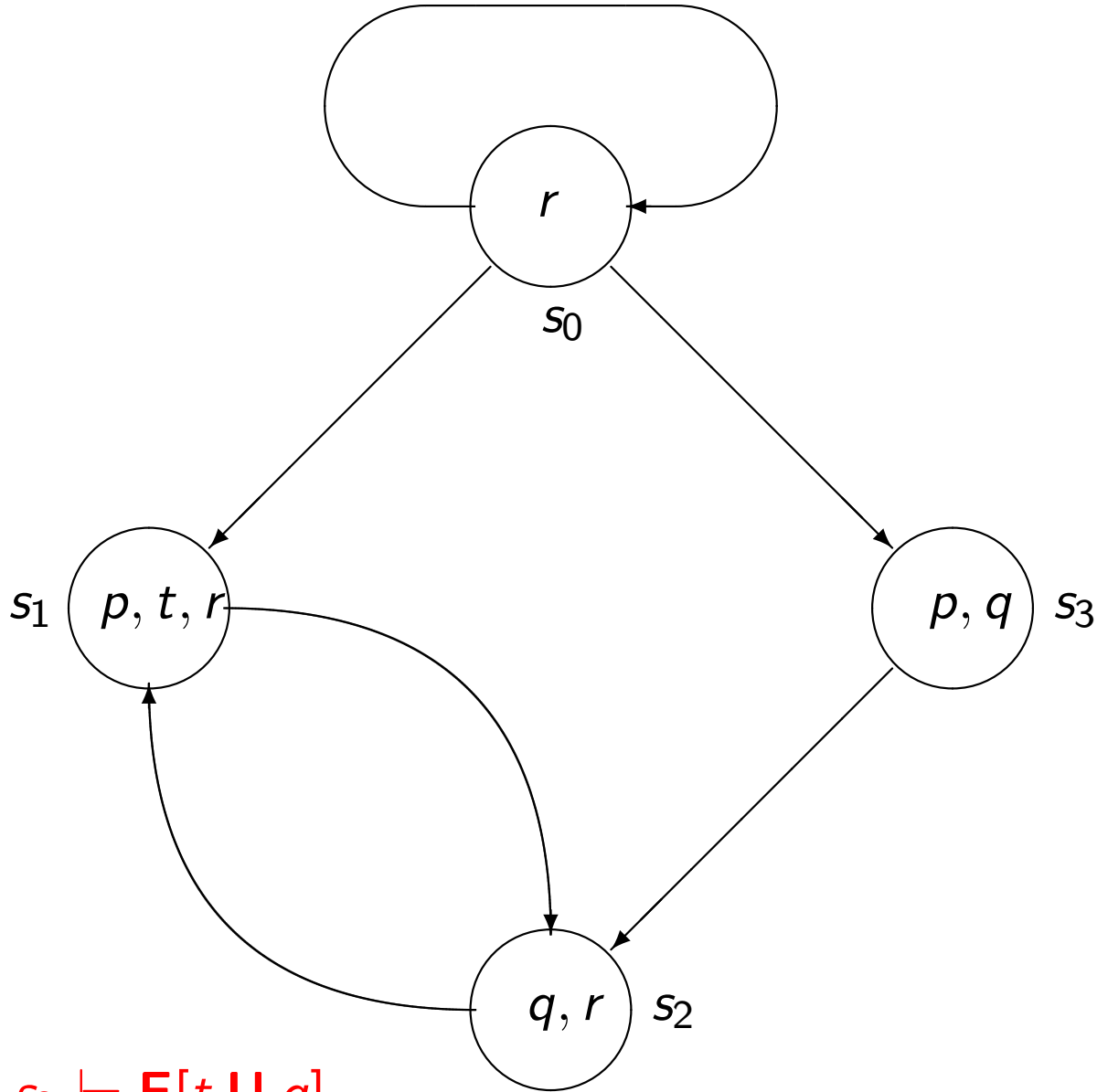
1. $\mathcal{M}_1, s_0 \models \mathbf{AF}\, t$

Проверьте выполнимость формул CTL в заданной модели.



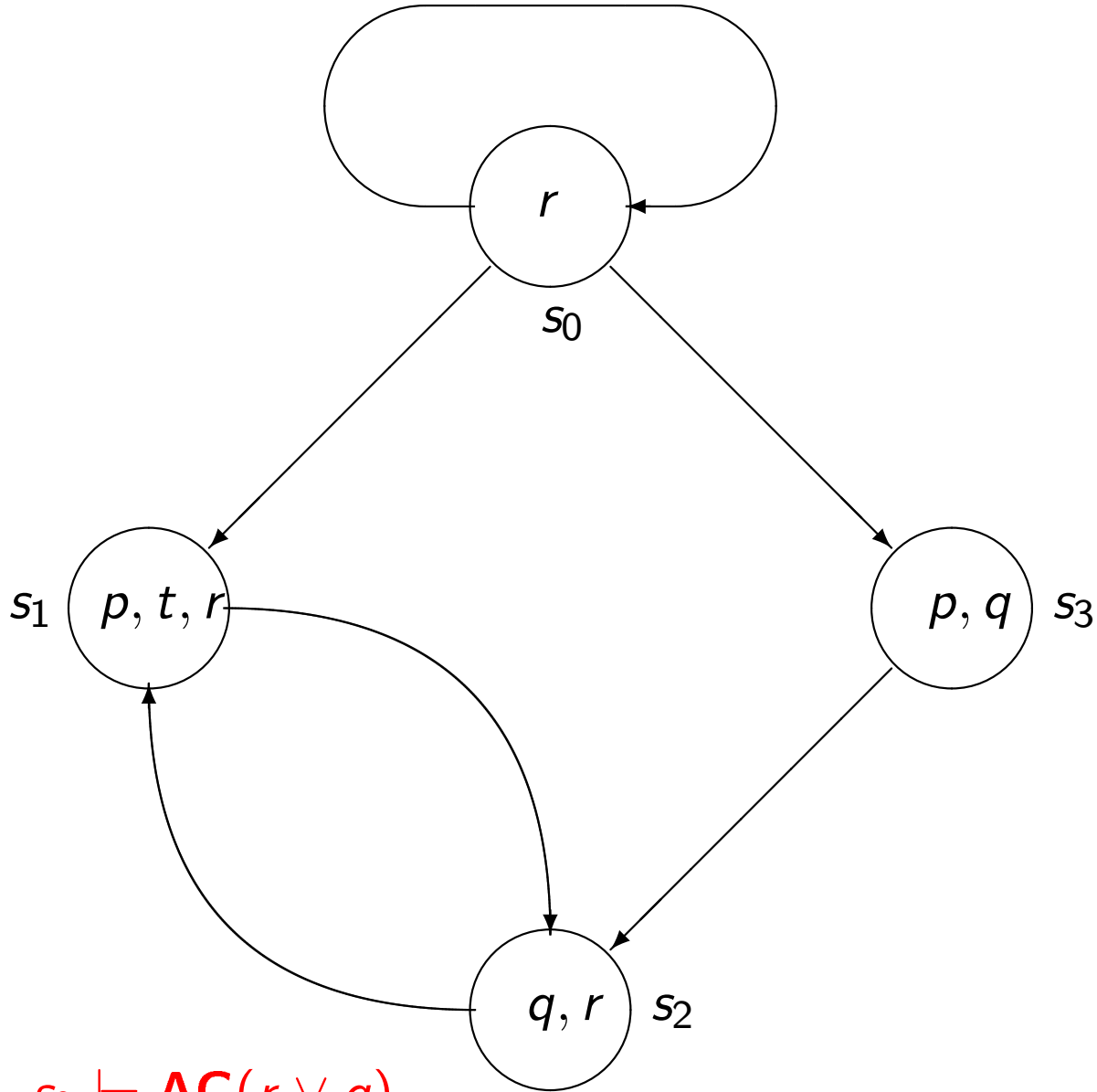
2. $\mathcal{M}_1, s_0 \models \mathbf{E}[t\ \mathbf{U}\ q]$

Проверьте выполнимость формул CTL в заданной модели.



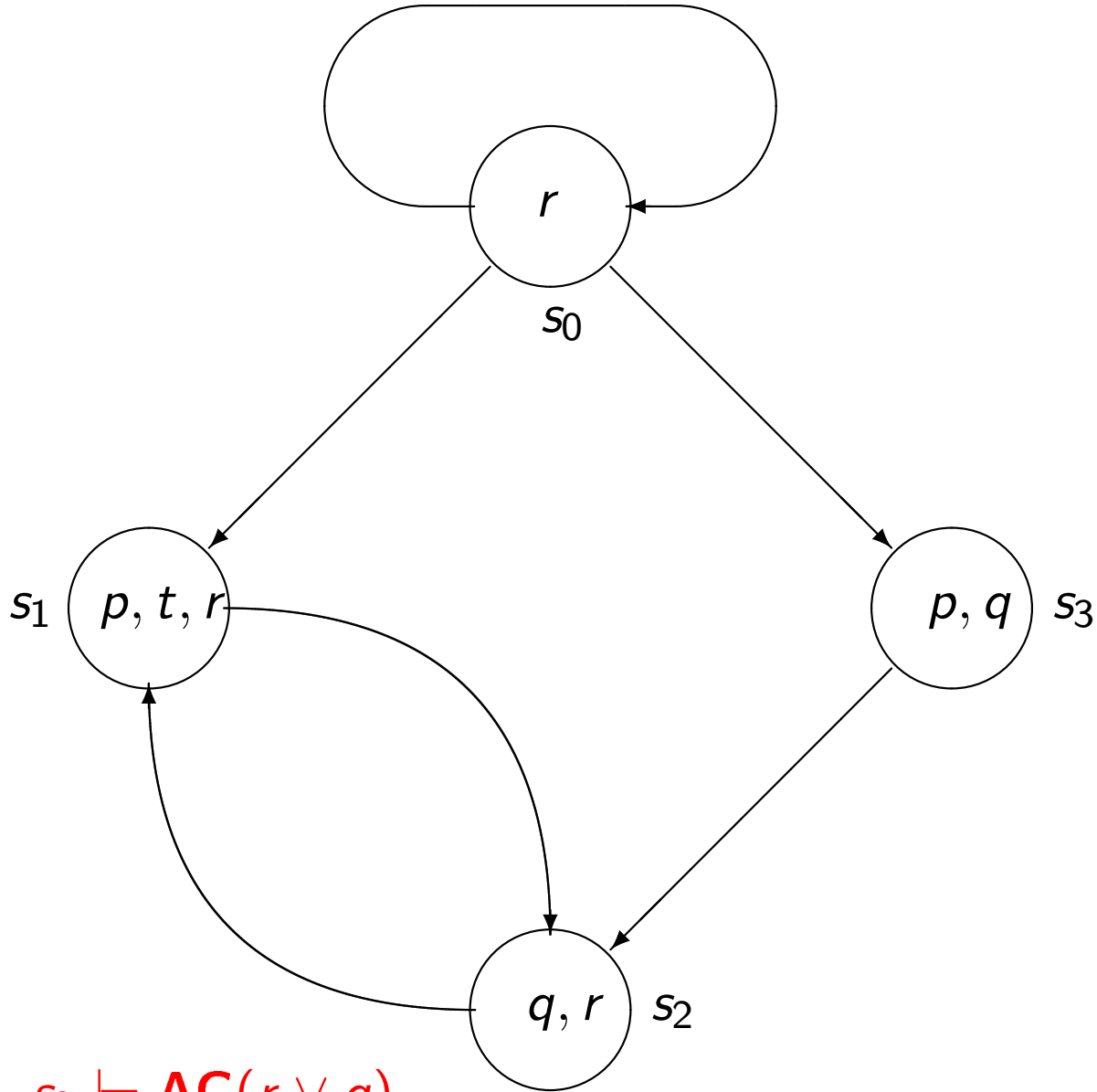
3. $\mathcal{M}_1, s_2 \models \mathbf{E}[t\ \mathbf{U}\ q]$

Проверьте выполнимость формул CTL в заданной модели.



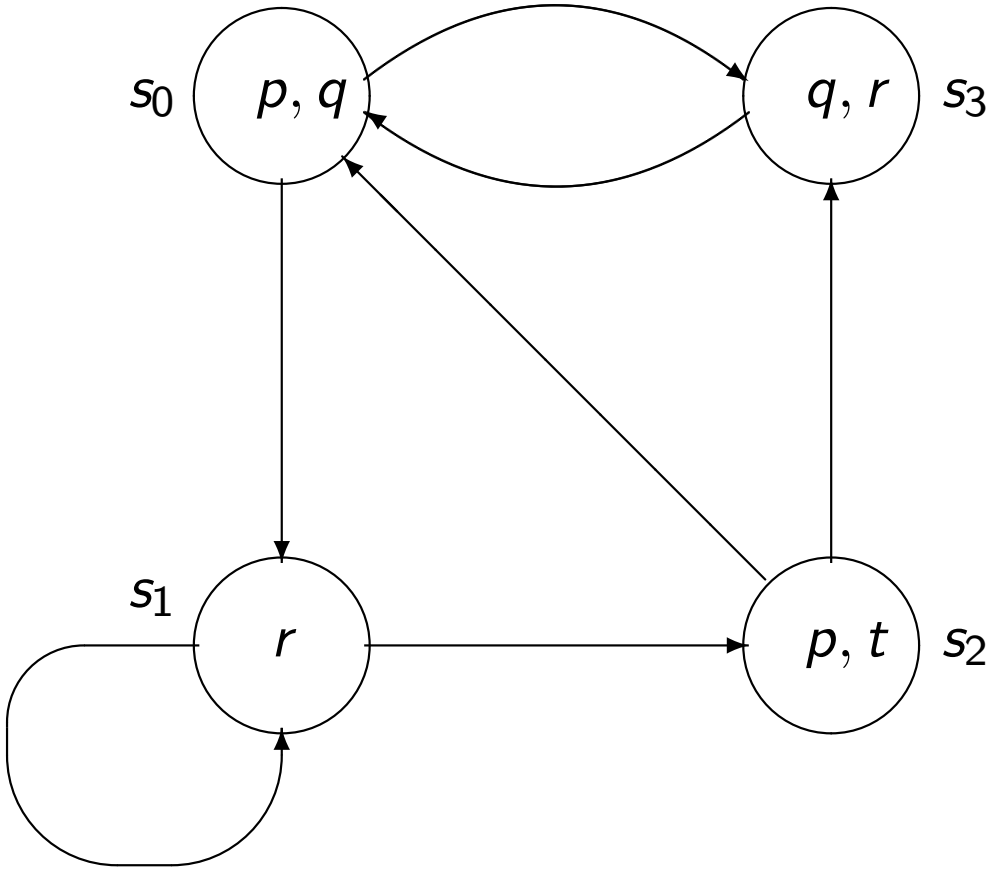
4. $\mathcal{M}_1, s_0 \models \mathbf{AG}(r \vee q)$

Проверьте выполнимость формул CTL в заданной модели.



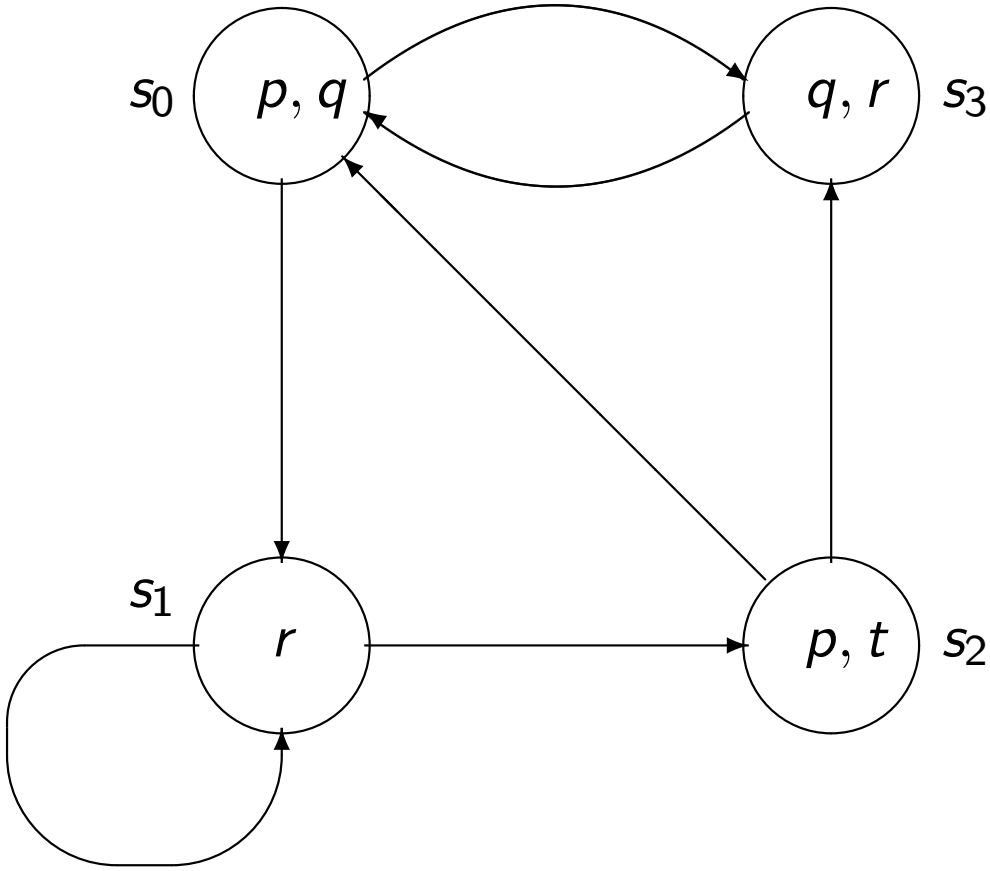
5. $\mathcal{M}_1, s_2 \models \mathbf{AG}(r \vee q)$

Проверьте выполнимость формул CTL в заданной модели.



6. $\mathcal{M}_1, s_0 \models \mathbf{AG}(\mathbf{AF}(p \vee r))$

Проверьте выполнимость формул CTL в заданной модели.



7. $\mathcal{M}_1, s_2 \models \mathbf{AG}(\mathbf{AF}(p \vee r))$

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.
- Всегда после получения запроса рано или поздно будет выдан ответ, если только запрос не будет отменен.

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.
- Всегда после получения запроса рано или поздно будет выдан ответ, если только запрос не будет отменен.
- Если вслед за событием p происходит событие q, то после этого событие r не наступит до тех пор, пока не случится событие t.

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.
- Всегда после получения запроса рано или поздно будет выдан ответ, если только запрос не будет отменен.
- Если вслед за событием p происходит событие q, то после этого событие r не наступит до тех пор, пока не случится событие t.
- Во всех вычислениях событие q предшествует событиям p и r.

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.
- Всегда после получения запроса рано или поздно будет выдан ответ, если только запрос не будет отменен.
- Если вслед за событием p происходит событие q, то после этого событие r не наступит до тех пор, пока не случится событие t.
- Во всех вычислениях событие q предшествует событиям p и r.
- Во всех вычислениях между событиями p и q ни разу не происходит события r.

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.
- Всегда после получения запроса рано или поздно будет выдан ответ, если только запрос не будет отменен.
- Если вслед за событием p происходит событие q, то после этого событие r не наступит до тех пор, пока не случится событие t.
- Во всех вычислениях событие q предшествует событиям p и r.
- Во всех вычислениях между событиями p и q ни разу не происходит события r.
- Во всех вычислениях событие p происходит не более двух раз.

Запишите формулы логик CTL, адекватно выражающие следующие суждения.

- Лифт, поднявшийся на второй этаж, может сколь угодно долго оставаться неподвижным.
- Всегда после получения запроса рано или поздно будет выдан ответ, если только запрос не будет отменен.
- Если вслед за событием p происходит событие q, то после этого событие r не наступит до тех пор, пока не случится событие t.
- Во всех вычислениях событие q предшествует событиям p и r.
- Во всех вычислениях между событиями p и q ни разу не происходит события r.
- Во всех вычислениях событие p происходит не более двух раз.
- Во всех вычислениях событие p происходит конечное число раз.

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;
- $\mathbf{EF}\,\varphi \vee \mathbf{EF}\,\psi$ и $\mathbf{EF}(\varphi \vee \psi)$;

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;
- $\mathbf{EF}\,\varphi \vee \mathbf{EF}\,\psi$ и $\mathbf{EF}(\varphi \vee \psi)$;
- $\mathbf{AF}\,\varphi \vee \mathbf{AF}\,\psi$ и $\mathbf{AF}(\varphi \vee \psi)$;

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;
- $\mathbf{EF}\,\varphi \vee \mathbf{EF}\,\psi$ и $\mathbf{EF}(\varphi \vee \psi)$;
- $\mathbf{AF}\,\varphi \vee \mathbf{AF}\,\psi$ и $\mathbf{AF}(\varphi \vee \psi)$;
- $AF\neg\varphi$ и $\neg\,\mathbf{EG}\,\varphi$;

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;
- $\mathbf{EF}\,\varphi \vee \mathbf{EF}\,\psi$ и $\mathbf{EF}(\varphi \vee \psi)$;
- $\mathbf{AF}\,\varphi \vee \mathbf{AF}\,\psi$ и $\mathbf{AF}(\varphi \vee \psi)$;
- $AF\neg\varphi$ и $\neg\,\mathbf{EG}\,\varphi$;
- $EF\neg\varphi$ и $\neg\,\mathbf{AF}\,\varphi$;

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;
- $\mathbf{EF}\,\varphi \vee \mathbf{EF}\,\psi$ и $\mathbf{EF}(\varphi \vee \psi)$;
- $\mathbf{AF}\,\varphi \vee \mathbf{AF}\,\psi$ и $\mathbf{AF}(\varphi \vee \psi)$;
- $AF\neg\varphi$ и $\neg\,\mathbf{EG}\,\varphi$;
- $EF\neg\varphi$ и $\neg\,\mathbf{AF}\,\varphi$;
- $\mathbf{A}[\varphi_1\ \mathbf{U}\ \mathbf{A}[\varphi_2\ \mathbf{U}\ \varphi_3]]$ и $\mathbf{A}[\mathbf{A}[\varphi_1\ \mathbf{U}\ \varphi_2]\ \mathbf{U}\ \varphi_3]$;

Какие из пар формул, приведенных ниже, являются эквивалентными?

- $\mathbf{EF}\,\varphi$ и $\mathbf{EG}\,\varphi$;
- $\mathbf{EF}\,\varphi \vee \mathbf{EF}\,\psi$ и $\mathbf{EF}(\varphi \vee \psi)$;
- $\mathbf{AF}\,\varphi \vee \mathbf{AF}\,\psi$ и $\mathbf{AF}(\varphi \vee \psi)$;
- $AF\neg\varphi$ и $\neg\,\mathbf{EG}\,\varphi$;
- $EF\neg\varphi$ и $\neg\,\mathbf{AF}\,\varphi$;
- $\mathbf{A}[\varphi_1\,\mathbf{U}\,\mathbf{A}[\varphi_2\,\mathbf{U}\,\varphi_3]]$ и $\mathbf{A}[\mathbf{A}[\varphi_1\,\mathbf{U}\,\varphi_2]\,\mathbf{U}\,\varphi_3]$;
- *true* и $\mathbf{AF}\,\varphi \;\rightarrow\; \mathbf{EG}\,\varphi$.

КОНЕЦ ЛЕКЦИИ 4.