Основы математической логики и логического программирования

ЛЕКТОР: В.А. Захаров

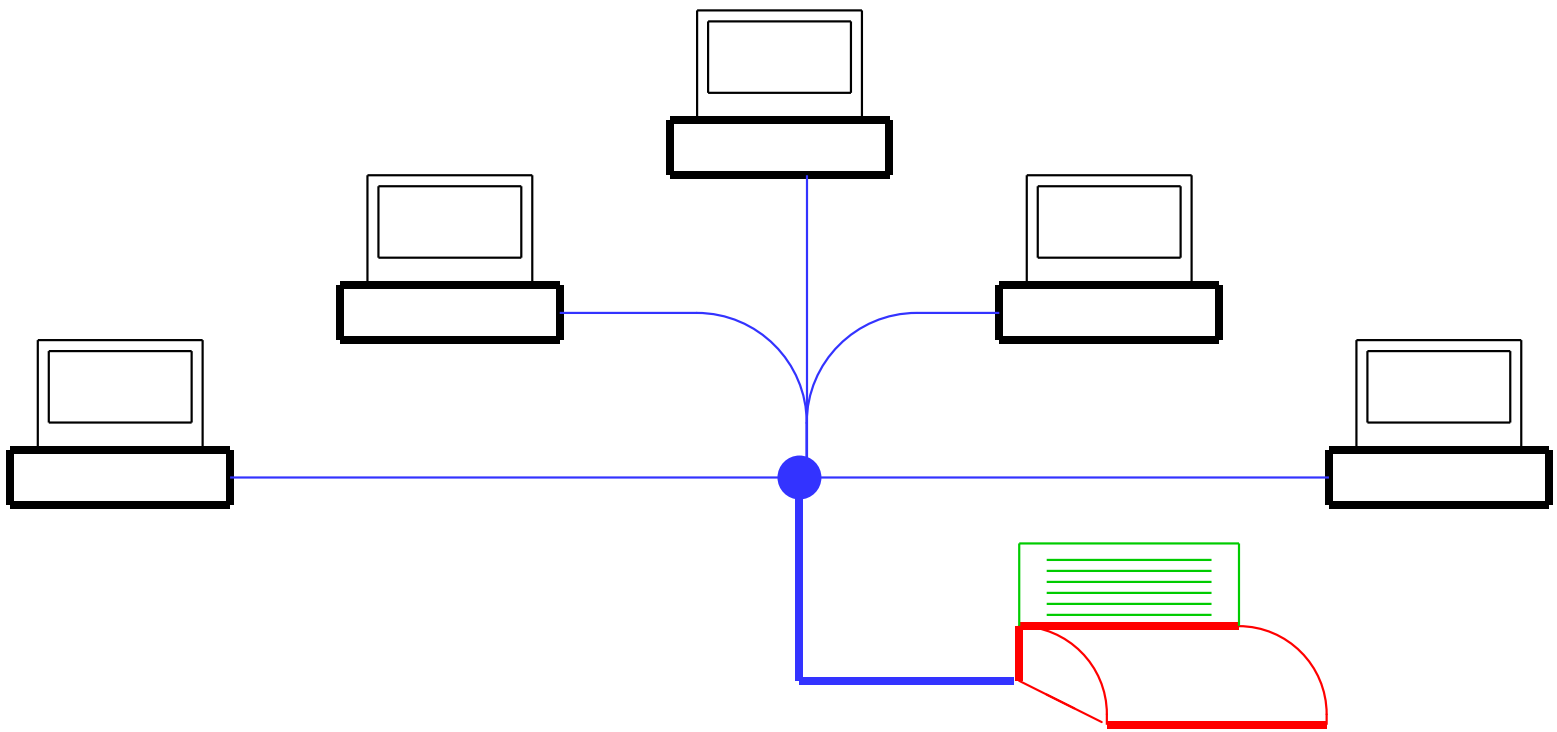
Лекция 21.

Верификация распределенных программ. Логика линейного времени PLTL. Размеченные системы переходов. Задача верификации моделей программ.

ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Задача

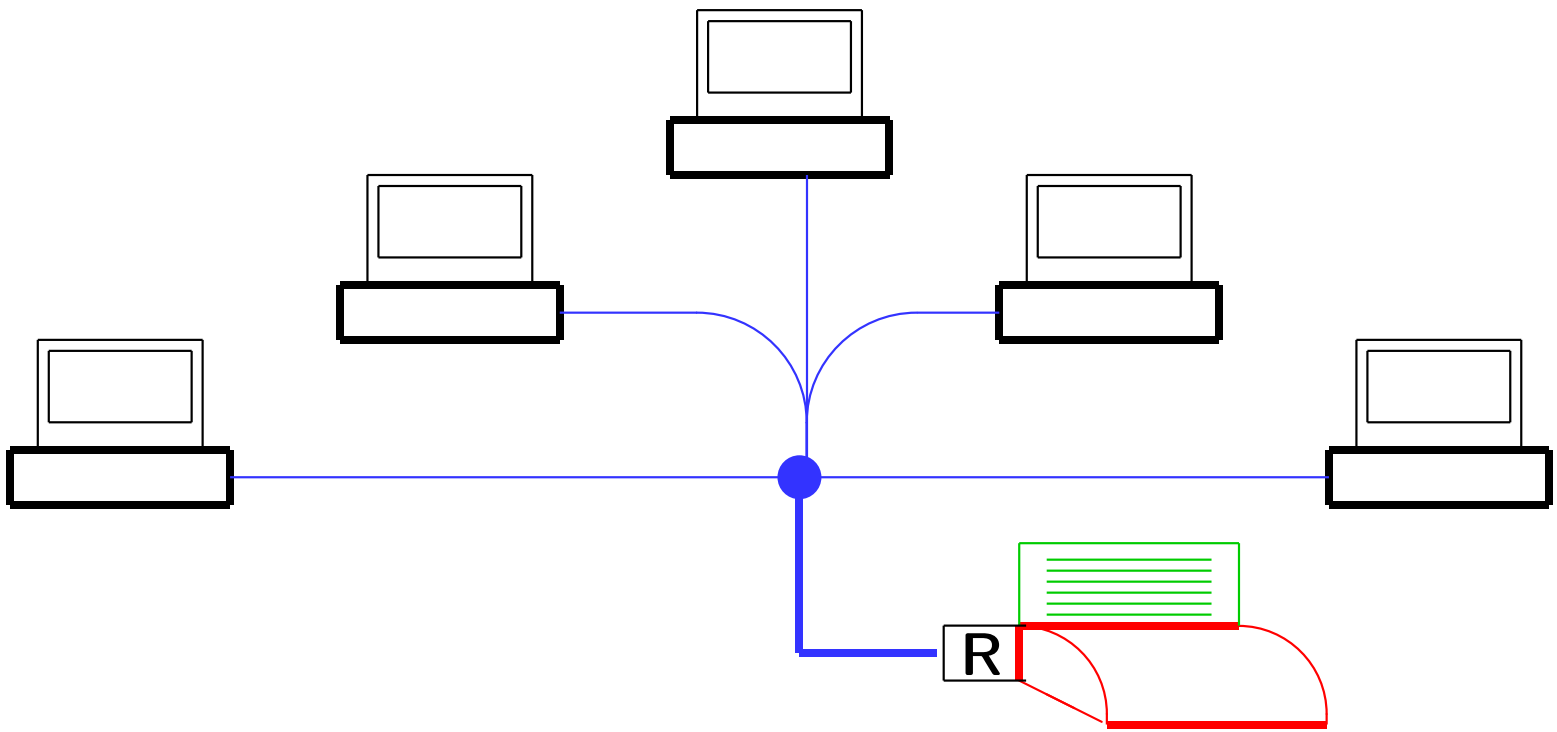
Имеется несколько компьютеров и только один принтер. Ни один компьютер не осведомлен о существовании других компьютеров. Как правильно организовать их взаимодействие, чтобы все они могли пользоваться этим принтером?



ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Задача

Предполагается также, что у принтера есть единственный однобитовый регистр **R**, общедоступный для считывания и записи. Этот регистр может находится в одном из двух состояний — *busy* (принтер занят) и *free* (принтер свободен).



ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Прежде чем писать программу (драйвер), обеспечивающую взаимодействие каждого компьютера с принтером, нужно сформулировать требования, предъявляемые к этой программе.

1. Всякий раз, когда принтер свободен и хотя бы один компьютер собирается отправить данные на печать, принтер будет рано или поздно занят;
2. Всякий раз, после того как принтер оказался занят, он должен когда-нибудь приступить к печати;
3. Компьютер, завершивший печать, должен когда-нибудь освободить принтер;
4. Данные на печать всегда передает не более чем один компьютер.

А какие еще требования разумно предъявить к нашему драйверу?

ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Для связи с принтером программист предложил снабдить каждый компьютер одной и той же программой

L_1: **while** $R \neq$ *free* **do wait od** ;

L_2: R = *busy*;

L_3: **output**(X,*printer*);

L_4: R = *free*;

Это простая и разумная программа.
Но будет ли система компьютеров, снабженных этой программой, вести себя в соответствии с указанными требованиями?

ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Если мы имеем дело с **последовательной программой**, то наиболее простой способ проверки ее правильности — это тестирование. Тестирование, вообще говоря, не гарантирует того, что все вычисления являются правильными, но оно позволяет, по крайней мере, убедиться в том, что программа ведет себя правильно на тестовых примерах.

Но если мы занимаемся верификацией **распределенных программ**, состоящих из нескольких процессов, работающих на независимых вычислительных устройствах, то тестирование не позволяет проверить правильность поведения распределенной системы даже на тестовых примерах.

Почему?

ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Предположим, что на заданных входных данных (тестовом примере) каждый из 20 процессов распределенной системы выполняет всего лишь одно действие. Тогда вычисление системы может быть физически реализовано $20! > 2^{18}$ способами в зависимости от той последовательности, в которой будут завершаться выполнения этих действий. Ясно, что рассмотреть все эти выполнения практически невозможно.

А вместе с тем одни те же действия, выполненные в разной последовательности, могут приводить к разным результатам:

«Наполнить бассейн водой» $\|$ «Прыгнуть в бассейн с вышки»

Как же проверять правильность распределенных систем?

ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ

Верификацию распределенных систем нужно автоматизировать. Это можно сделать, например, так.

1. Выбрать логический язык $\mathcal{L}$, на котором можно описывать требования, предъявляемые к программе. Представить эти требования в виде формул $\varphi_1, \ldots, \varphi_n$.
2. Выбрать математическую модель M, адекватно представляющую все вычисления программы. Модель должна быть устроен так, чтобы каждое вычисление I в модели M являлось интерпретацией языка $\mathcal{L}$.
3. Проверить выполнимость формул $\varphi_1, \ldots, \varphi_n$ на всех вычислениях модели M. Для проверки выполнимости формул языка $\mathcal{L}$ на модели программы M должен быть разработан эффективный алгоритм.

Такой подход к проверке правильности программ назвается верификацией моделей программ (англ. model-checking).

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

При верификации распределенных систем, как правило, требуется проверить, что в каждом вычислении системы некоторые события (выполнение того или иного действия, прием/передача сообщений и пр.) происходят в определенной последовательности.

Каждое событие *event* можно охарактеризовать булевой переменной (0-местным предикатом) p_{event}, которая принимает значение **true** в том и только том случае, когда осуществляется событие *event*. Таким образом, в логическом языке $\mathcal{L}$ не нужны предметные переменные, термы, кванторы.

Однако осуществимость событий (значения булевых пременных p_{event}) изменяется со временем. Значит, в логическом языке $\mathcal{L}$ должен быть явно учтен феномен времени.

Таким образом, для описания требований, которые предъявляются к распределенной системе, достаточно воспользоваться языком пропозициональной темпоральной логики линейного времени (PLTL).

Исторические сведения

1941

АМИР ПНУЭЛИ

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Синтаксис PLTL

В PLTL наряду с булевыми логическими связками для описания причинно-следственной зависимости событий во времени применяются темпоральные операторы

- **X** (ne**X**ttime) «в следующий момент времени»;
- **F** (sometime in **F**uture) «когда-то в будущем»;
- **G** (**G**lobally) «всегда в будущем»;
- **U** (**U**ntil) «до тех пор пока»;
- **R** (**R**elease) «высвободить, открепить».

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Пусть задано множество булевых переменных $\mathcal{AP} = \{p_1, p, \ldots, p_n, \ldots\}$ (будем называть их атомарными высказываниями).

Синтаксис PLTL

Формула PLTL — это

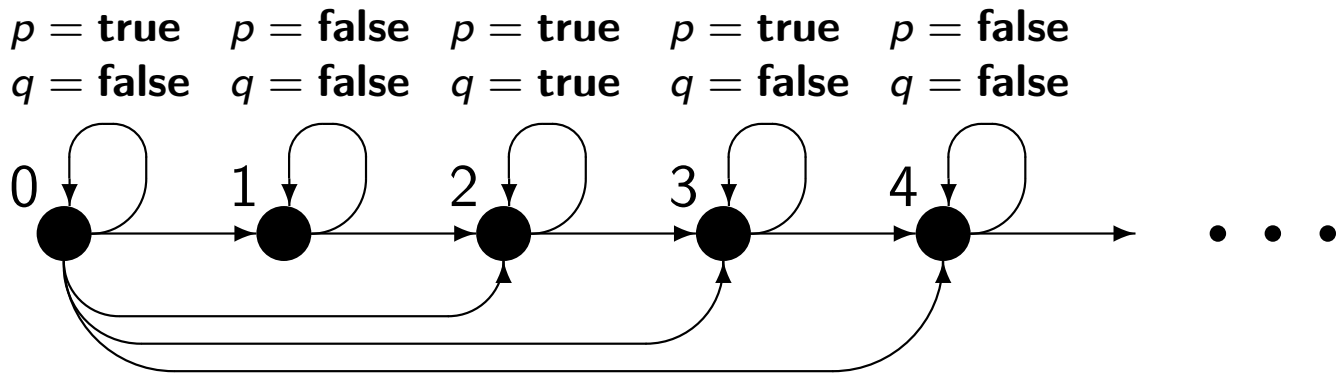
p_i,	если $p_i \in \mathcal{AP}$;
$(\varphi \& \psi)$,	если φ и ψ — формулы;
$(\varphi \vee \psi)$,	
$(\varphi \to \psi)$,	
$(\neg\varphi)$,	
$(\mathbf{X}\varphi)$,	«в следующий момент будет верно φ»;
$(\mathbf{F}\varphi)$,	«когда-то в будущем будет верно φ»;
$(\mathbf{G}\varphi)$,	«всегда верно φ»;
$(\varphi \mathbf{U} \psi)$,	«φ остается верной, пока не станет верной ψ»;
$(\varphi \mathbf{R} \psi)$,	«ψ может перестать быть верной только после того, как станет верной φ».

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Семантика PLTL

Интерпретация PLTL — это темпоральная модель Крипке $I = \langle \mathbb{N}, \leq, \xi \rangle$, где

- $\mathbb{N} = \{0, 1, 2, \dots\}$ — множество моментов времени;
- $\leq$ — отношение нестрогого линейного порядка на $\mathbb{N}$;
- $\xi : \mathbb{N} \times \mathcal{AP} \to \{\mathbf{true}, \mathbf{false}\}$ — оценка атомарных высказываний на шкале времени.



ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Семантика PLTL

Интерпретация $I = \langle \mathbb{N}, \leq, \xi \rangle$ — это вычислительная трасса программы, и в этой трассе

- $\mathbb{N} = \{0, 1, 2, \dots\}$ — это последовательность состояний вычисления, линейно упорядоченная отношением переходов $\leq$;
- оценка $\xi : \mathbb{N} \times \mathcal{AP} \to \{\textbf{true}, \textbf{false}\}$ указывает, какие события происходят в те или иные моменты времени.

Формулы PLTL — это утверждения о том, в какой последовательности должны происходить события по ходу вычислений программ.

Чтобы оценивать, в какой мере вычислительная трасса (интерпретация) удовлетворяет заданному требованию (формуле PLTL), определим отношение выполнимости формул PLTL в темпоральных интерпретациях.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

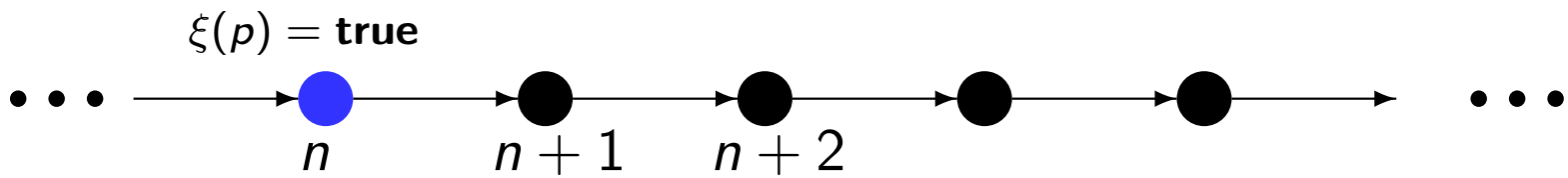
Семантика PLTL

Пусть $I = \langle \mathbb{N}, \leq, \xi \rangle$ — темпоральная интерпретация (вычислительная трасса), $n \in \mathbb{N}$ — момент времени (состояние вычисления), φ — формула PLTL.

Тогда отношение выполнимости $I, n \models \varphi$ формулы φ в момент времени n в интерпретации I определяется так.

1. Если $\varphi = p$, $p \in \mathcal{AP}$ (т. е. φ — атомарное высказывание), то

$$I, n \models \varphi \iff \xi(p) = \textbf{true}.$$

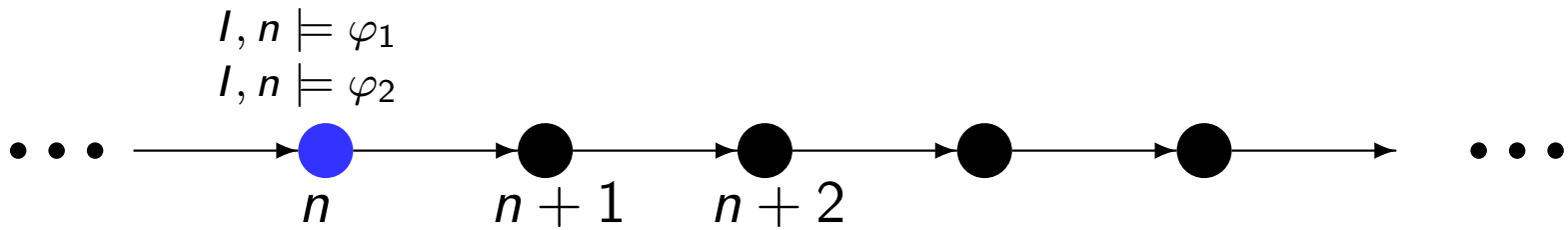


ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Семантика PLTL

2. Если $\varphi = \varphi_1 \& \varphi_2$, то

$$I, n \models \varphi \iff I, n \models \varphi_1 \text{ и } I, n \models \varphi_2.$$



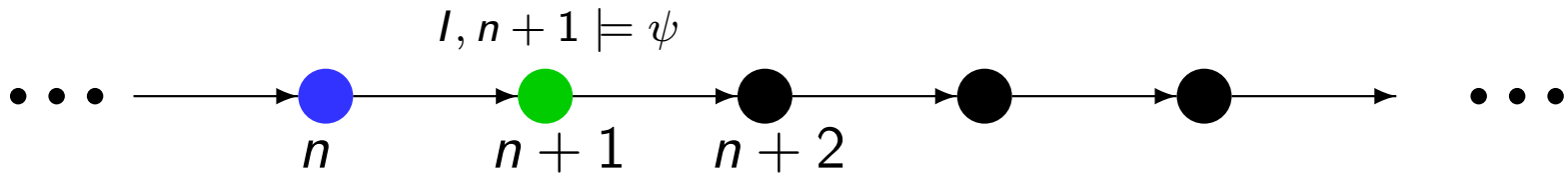
Для формул вида $\varphi_1 \vee \varphi_2$, $\varphi_1 \to \varphi_2$, $\neg\varphi_1$ отношение выполнимости в темпоральной модели определяется точно так же, как в классической логике предикатов.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Семантика PLTL

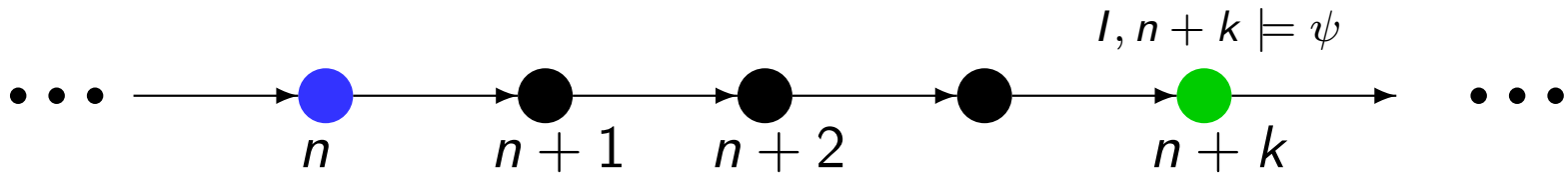
3. Если $\varphi = \mathbf{X}\psi$, то

$$l, n \models \varphi \iff l, n+1 \models \psi.$$



4. Если $\varphi = \mathbf{F}\psi$, то

$l, n \models \varphi \iff$ существует такое $k, k \geq 0,$ что $l, n+k \models \psi.$

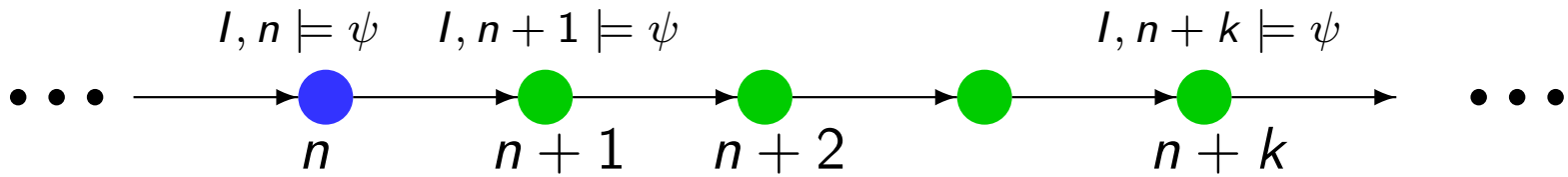


ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Семантика PLTL

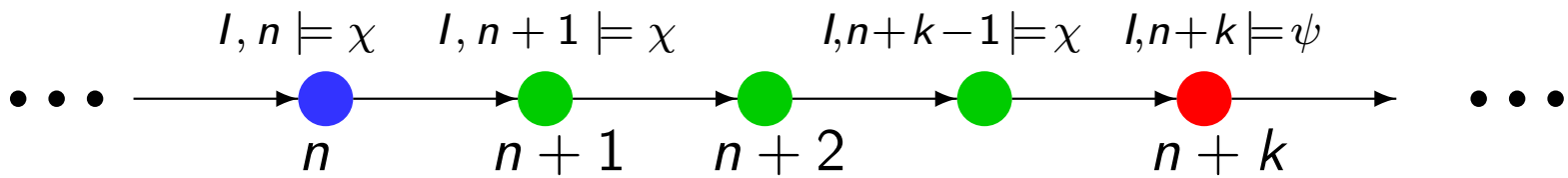
5. Если $\varphi = \mathbf{G}\psi$, то

$$I, n \models \varphi \iff \text{для любого } k, k \geq 0, \text{ верно } I, n+k \models \psi.$$



6. Если $\varphi = \chi\mathbf{U}\psi$, то

$$I, n \models \varphi \iff \text{существует такое } k, k \geq 0, \text{ что } I, n+k \models \psi,$$
$$\text{и для любого } i, 0 \leq i < k, \text{ верно } I, n+i \models \chi.$$



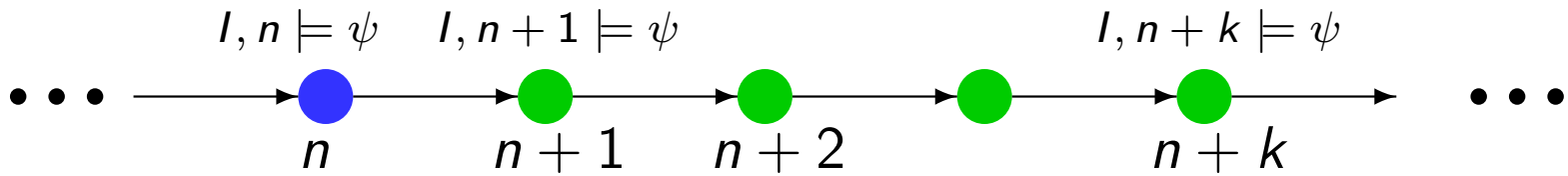
ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Семантика PLTL

7. Если $\varphi = \chi \mathbf{R} \psi$, то

$$I, n \models \varphi \iff \text{либо для любого } k, k \geq 0, \text{ верно } I, n+k \models \psi,$$

,



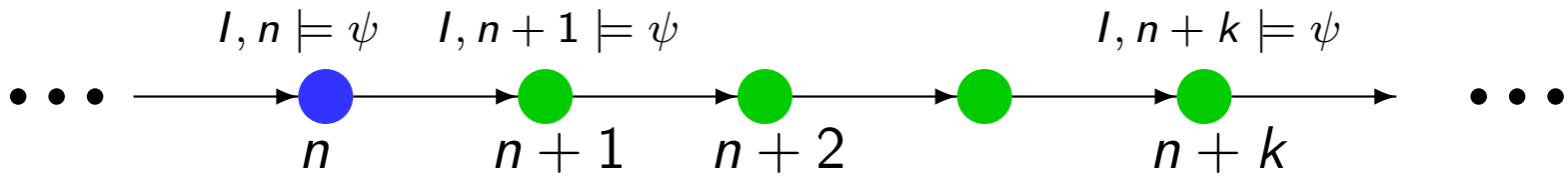
ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

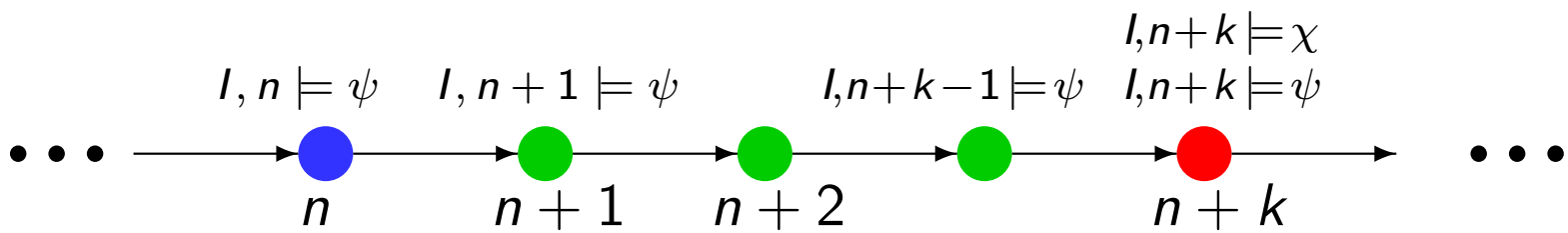
Семантика PLTL

7. Если $\varphi = \chi \mathbf{R} \psi$, то

$$I, n \models \varphi \iff$$ либо для любого $k, k \geq 0$, верно $I, n + k \models \psi$,

либо существует такое $k, k \geq 0$, что $I, n + k \models \chi$,
и для любого $i, 0 \leq i \leq k$, верно $I, n + i \models \psi$.





ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Будем называть формулу PLTL φ

- выполнимой в интерпретации I, если верно $I, 0 \models \varphi$ (обозначается $I \models \varphi$);
- PLTL-общезначимой , если для любой интерпретации I верно $I \models \varphi$ (обозначается $\models \varphi$).

Чтобы облегчить запись формул и избавиться от лишних скобок, условимся, что
одноместные темпоральные операторы **X**, **F**, **G** обладают таким же приоритетом, как отрицание $\neg$,
а двухместные темпоральные операторы **U**, **R** имеют наивысший приоритет среди двухместных связок.

Таким образом, запись

$$\mathbf{X}p_1\mathbf{U}p_2\&\mathbf{F}p_3 \rightarrow \neg p_1\mathbf{R}p_2$$

обозначает формулу

$$\big(((\mathbf{X}p_1)\mathbf{U}p_2)\&(\mathbf{F}p_3)\big) \rightarrow \big((\neg p_1)\mathbf{R}p_2\big).$$

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Равносильные формулы

Темпоральные операторы PLTL связаны друг с другом определенными соотношениями (равносильностями). Вот наиболее важные из соотношений равносильности.

Законы двойственности.

1. $\models \neg \mathbf{X}\varphi \equiv \mathbf{X}\neg\varphi$;
2. $\models \neg \mathbf{F}\varphi \equiv \mathbf{G}\neg\varphi$;
3. $\models \neg \mathbf{G}\varphi \equiv \mathbf{F}\neg\varphi$;
4. $\models \neg(\varphi \mathbf{U} \psi) \equiv \neg\varphi \mathbf{R} \neg\psi$;
5. $\models \neg(\varphi \mathbf{R} \psi) \equiv \neg\varphi \mathbf{U} \neg\psi$.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Равносильные формулы

Темпоральные операторы PLTL связаны друг с другом определенными соотношениями (равносильностями).
Вот наиболее важные из соотношений равносильности.

Законы двойственности.

1. $\models \neg \mathbf{X}\varphi \equiv \mathbf{X}\neg\varphi$;
2. $\models \neg \mathbf{F}\varphi \equiv \mathbf{G}\neg\varphi$;
3. $\models \neg \mathbf{G}\varphi \equiv \mathbf{F}\neg\varphi$;
4. $\models \neg(\varphi \mathbf{U} \psi) \equiv \neg\varphi \mathbf{R} \neg\psi$;
5. $\models \neg(\varphi \mathbf{R} \psi) \equiv \neg\varphi \mathbf{U} \neg\psi$.

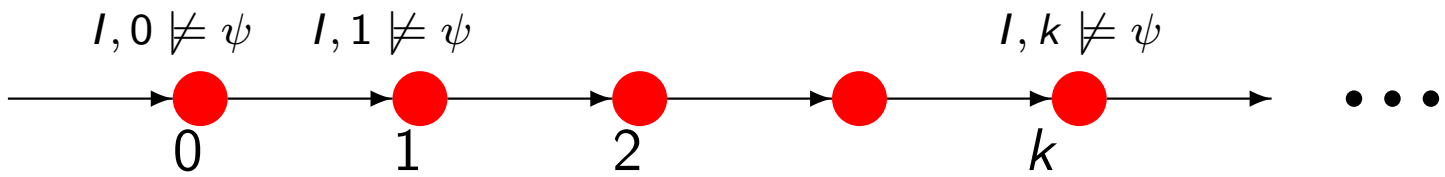
Доказательство. Следует непосредственно из определения отношения выполнимости.
Покажем справедливость соотношения 4).

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

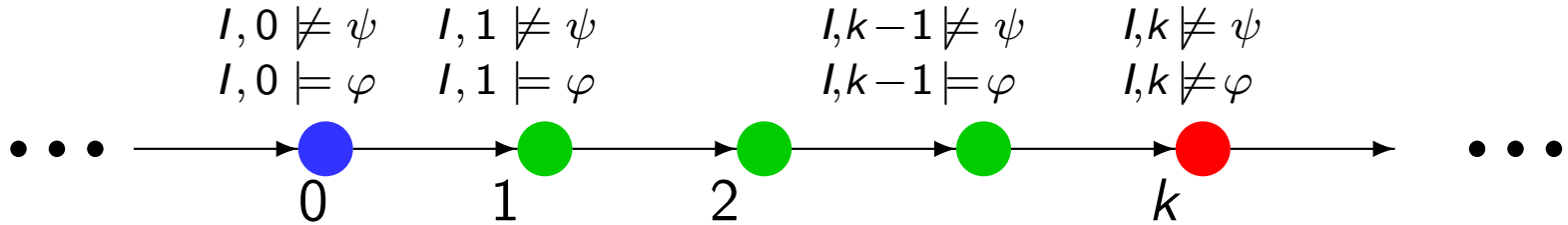
Доказательство.

Пусть $I, 0 \models \neg(\varphi \mathbf{U} \psi)$, т. е. $I, 0 \not\models \varphi \mathbf{U} \psi$. Тогда согласно определению отношения выполнимости для оператора **U** верно хотя бы одно из двух:

1. либо для любого k, $k \geq 0$, верно $I, k \not\models \psi$



2. либо существует такое k, $k \geq 0$, что $I, k \not\models \psi$, $I, k \not\models \varphi$ и при этом для любого i если $0 \leq i < k$, то $I, k \not\models \psi$, $I, k \models \varphi$

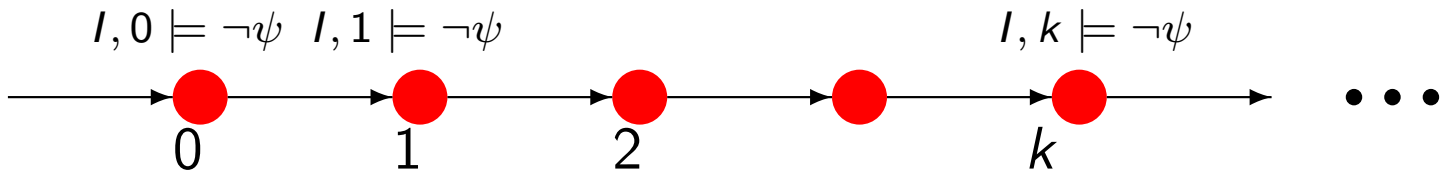


ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

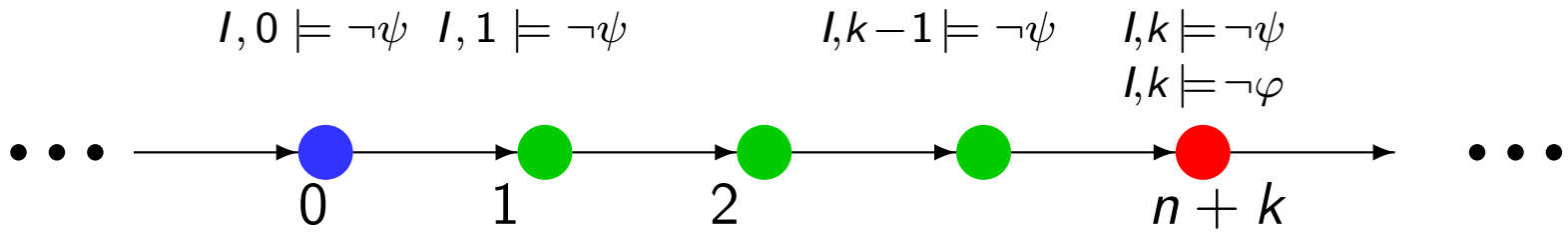
Доказательство.

Но это означает, что верно хотя бы одно из двух:

1. либо для любого k, $k \geq 0$, верно $I, k \models \neg\psi$



2. либо существует такое k, $k \geq 0$, что $I, k \models \neg\psi$, $I, k \models \neg\varphi$ и при этом для любого i если $0 \leq i < k$, то $I, k \models \neg\psi$



А это как раз и означает, что $I, 0 \models (\neg\varphi)\mathbf{R}(\neg\psi)$.

Значит, $I \models \neg(\varphi\mathbf{U}\psi) \rightarrow \neg\varphi\mathbf{R}\neg\psi$ для любой интерпретации I.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Доказательство.

Проводя аналогичные рассуждения покажите самостоятельно , что верно соотношение

$$I \models \neg\varphi \mathbf{R} \neg\psi \rightarrow \neg(\varphi \mathbf{U} \psi),$$

а также остальные законы двойственности темпоральных операторов.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Равносильные формулы

Законы взаимной зависимости.

1. $\models \mathbf{F}\varphi \equiv \neg\mathbf{G}\neg\varphi$;
2. $\models \mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$;
3. $\models \varphi\mathbf{U}\psi \equiv \neg(\neg\varphi\mathbf{R}\neg\psi)$;
4. $\models \varphi\mathbf{R}\psi \equiv \neg(\neg\varphi\mathbf{U}\neg\psi)$;
5. $\models \mathbf{F}\varphi \equiv \mathbf{true}\ \mathbf{U}\varphi$;
6. $\models \mathbf{G}\varphi \equiv \mathbf{false}\ \mathbf{R}\varphi$.

Доказательство. Самостоятельно.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Равносильные формулы

Законы неподвижной точки.

1. $\models \mathbf{F}\varphi \equiv \varphi \vee \mathbf{XF}\varphi$;
2. $\models \mathbf{G}\varphi \equiv \varphi \,\&\, \mathbf{XG}\varphi$;
3. $\models \varphi\mathbf{U}\psi \equiv \psi \vee (\varphi \,\&\, \mathbf{X}(\varphi\mathbf{U}\psi)$;
4. $\models \varphi\mathbf{R}\psi \equiv \psi \,\&\, (\varphi \vee \mathbf{X}(\varphi\mathbf{R}\psi)$.

Доказательство. Самостоятельно.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Равносильные формулы

А какие законы дистрибутивности верны?

1. $\models \mathbf{F}(\varphi \vee \psi) \equiv$???;
2. $\models \mathbf{G}(\varphi \vee \psi) \equiv$???;
3. $\models \mathbf{F}(\varphi \& \psi) \equiv$???;
4. $\models \mathbf{G}(\varphi \& \psi) \equiv$???;
5. $\models \varphi \mathbf{U}(\psi \vee \chi) \equiv$???;
6. $\models \varphi \mathbf{U}(\psi \& \chi) \equiv$???
7. $\models (\varphi \vee \chi)\mathbf{U}\chi \equiv$???;
8. $\models (\varphi \& \chi)\mathbf{U}\chi \equiv$???.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Выразительные возможности PLTL

А теперь посмотрим, насколько адекватно и просто можно выражать при помощи PLTL те требования, которые предъявляются к поведению распределенных систем программ.

1. Всякий раз, когда принтер свободен и хотя бы один компьютер собирается отправить данные на печать, принтер будет рано или поздно занят;
2. Всякий раз, после того как принтер оказался занят, он должен когда-нибудь приступить к печати;
3. Компьютер, завершивший печать, должен когда-нибудь освободить принтер;
4. Данные на печать всегда передает не более чем один компьютер.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Выразительные возможности PLTL

Введем следующие атомарные высказывания, соответствующие основным событиям вычислений программы:

1. try_i — i-ый компьютер собирается отправить данные на печать;
2. pr_i — i-ый компьютер передает данные на печать;
3. *free* — принтер свободен;
4. *busy* — принтер занят.

Пусть имеется система, состоящая из 2 компьютеров.
Тогда все перечисленные требования выражаются формулами PLTL так.

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Выразительные возможности PLTL

1. Всякий раз, когда принтер свободен и хотя бы один компьютер собирается отправить данные на печать, принтер будет рано или поздно занят:

$$\varphi_1 = \mathbf{G}((\mathit{free} \;\&\; (\mathit{try}_1 \;\vee\; \mathit{try}_2)) \;\rightarrow\; \mathbf{F}\mathit{busy}) \;;$$

2. Всякий раз, после того как принтер оказался занят, он должен когда-нибудь приступить к печати:

$$\varphi_2 = \mathbf{G}(\mathit{free} \;\&\; \mathbf{X}\mathit{busy} \;\rightarrow\; \mathbf{XF}(\mathit{pr}_1 \;\vee\; \mathit{pr}_2)) \;;$$

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Выразительные возможности PLTL

3. Компьютер, завершивший печать, должен когда-нибудь освободить принтер:

$$\varphi_{3i} = \mathbf{G}(pr_i \ \& \ \mathbf{X}\neg pr_i \ \rightarrow \ \mathbf{XF}free) \ ;$$

4. Данные на печать всегда передает не более чем один компьютер:

$$\varphi_4 = \mathbf{G}(\neg(pr_1 \ \& \ pr_2)) \ .$$

ЛОГИКА ЛИНЕЙНОГО ВРЕМЕНИ PLTL

Выразительные возможности PLTL

5. До тех пор пока хотя бы один компьютер отправляет данные на печать, принтер остается занятым:

$$\varphi_5 = \mathbf{G}((\neg(pr_1 \ \vee \ pr_2) \ \mathbf{R} \ busy) \ ;$$

или может быть это лучше выразить так:

$$\varphi_5' = \mathbf{G}(busy \ \mathbf{R} \ (\neg(pr_1 \ \vee \ pr_2))) \ ?$$

или может быть так:

$$\varphi_5'' = \mathbf{G}((pr_1 \ \vee \ pr_2) \ \rightarrow \ busy) \ ?$$

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

Теперь мы можем использовать PLTL в качестве формального языка спецификации программ.
Чтобы иметь возможность проверить, удовлетворяют ли все вычисления распределенной системы программ заданным спецификациям, которые представлены формулами PLTL, нужно определить математическую модель программ.

При разработке математической модели программ нужно стремиться к тому, чтобы

- каждое вычисление распределенной системы программ представляло собой темпоральную интерпретацию;
- вычисления программ в предложенной математической модели соответствовали реальному поведению вычислительных устройств, выполняющих эти программы;
- модель программ имела простое устройство.

В качестве такой модели программ мы будем использовать размеченные системы переходов .

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

Определение LTS

Размеченная система переходов (LTS, Labelled Transition System) — это пятерка $\langle \mathcal{AP}, S, S_0, \longrightarrow, \rho \rangle$, в которой

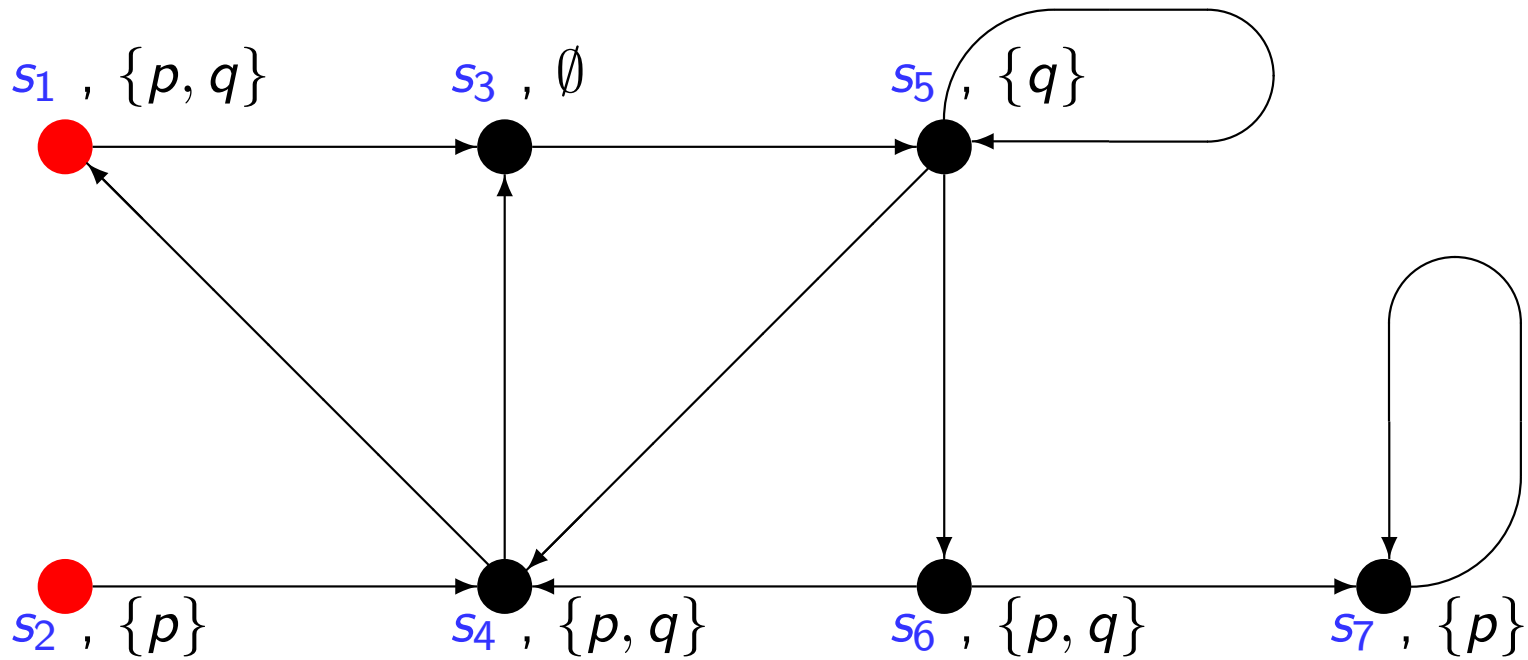
1. $\mathcal{AP}$ — множество атомарных высказываний;
2. S — непустое множество состояний вычислений;
3. S_0, $S_0 \subseteq S$, — непустое подмножество начальных состояний;
4. $\longrightarrow \subseteq S \times S$, — **тотальное** отношение переходов, тотальность отношения $\longrightarrow$ означает, что для любого состояния $s, s \in S$, существует такое состояние s', что $s \longrightarrow s'$ (т. е. из любого состояния можно сделать хотя бы один переход);
5. $\rho : S \rightarrow 2^{\mathcal{AP}}$ — функция разметки, приписывающая каждому состоянию вычислений $s, s \in S$, множество $\rho(s)$, $\rho(s) \subseteq \mathcal{AP}$, всех тех атомарных высказываний, которые являются истинными в состоянии s.

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

Пример LTS

$$M = \langle \mathcal{AP}, S, S_0, T, \rho \rangle$$

$$\mathcal{AP} = \{p, q\}, \quad S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}, \quad S_0 = \{s_1, s_2\}$$



РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS и PLTL-интерпретации

Трассой в LTS $M = \langle \mathcal{AP}, S, S_0, \longrightarrow, \rho \rangle$ называется всякая бесконечная последовательность состояний

$$tr = s_{i_0}, s_{i_1}, \ldots, s_{i_n}, s_{i_{n+1}}, \ldots, \qquad (*)$$

в которой для любого $n, n > 0$, верно $(s_{i_n} \longrightarrow s_{i_{n+1}})$.

Если s_{i_0} — начальное состояние, $s_{i_0} \in S_0$, то трасса tr называется начальной трассой.

Запись $Tr(M)$ обозначает множество всех трасс LTS M, а запись $Tr_0(M)$ — множество всех начальных трасс LTS M.

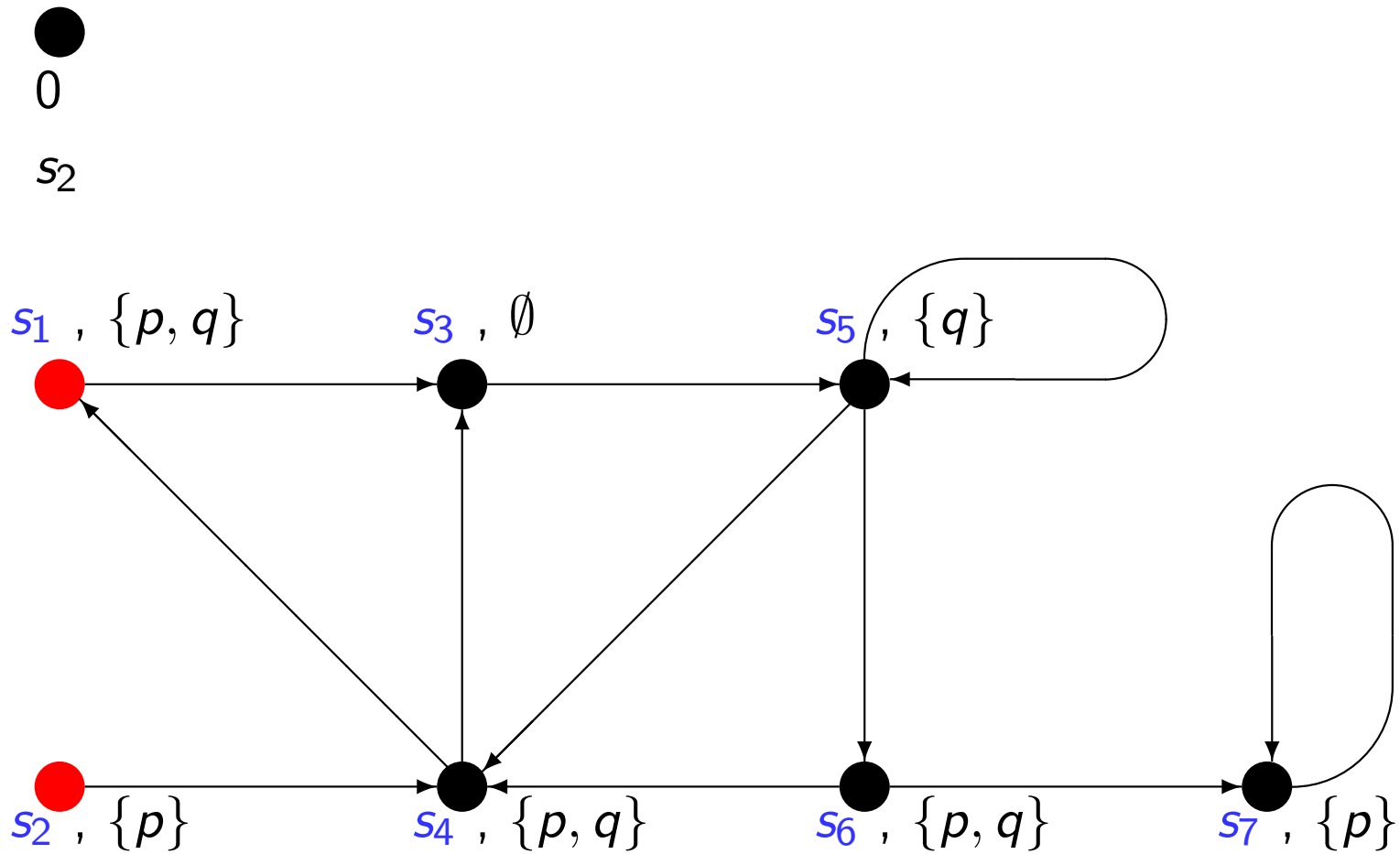
Каждой трассе $tr \in Tr(M)$ вида $(*)$ сопоставим PLTL-интерпретацию $I(tr) = \langle \mathbb{N}, \leq, \xi \rangle$, в которой для любого $n, n \geq 0$, и p, $p \in \mathcal{AP}$, верно соотношение

$$\xi(n, p) = \textbf{true} \iff p \in \rho(s_{i_n}) \ .$$

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

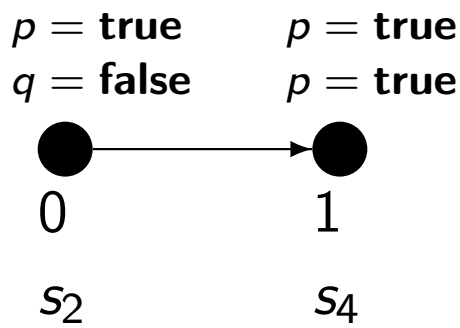
Пример LTS

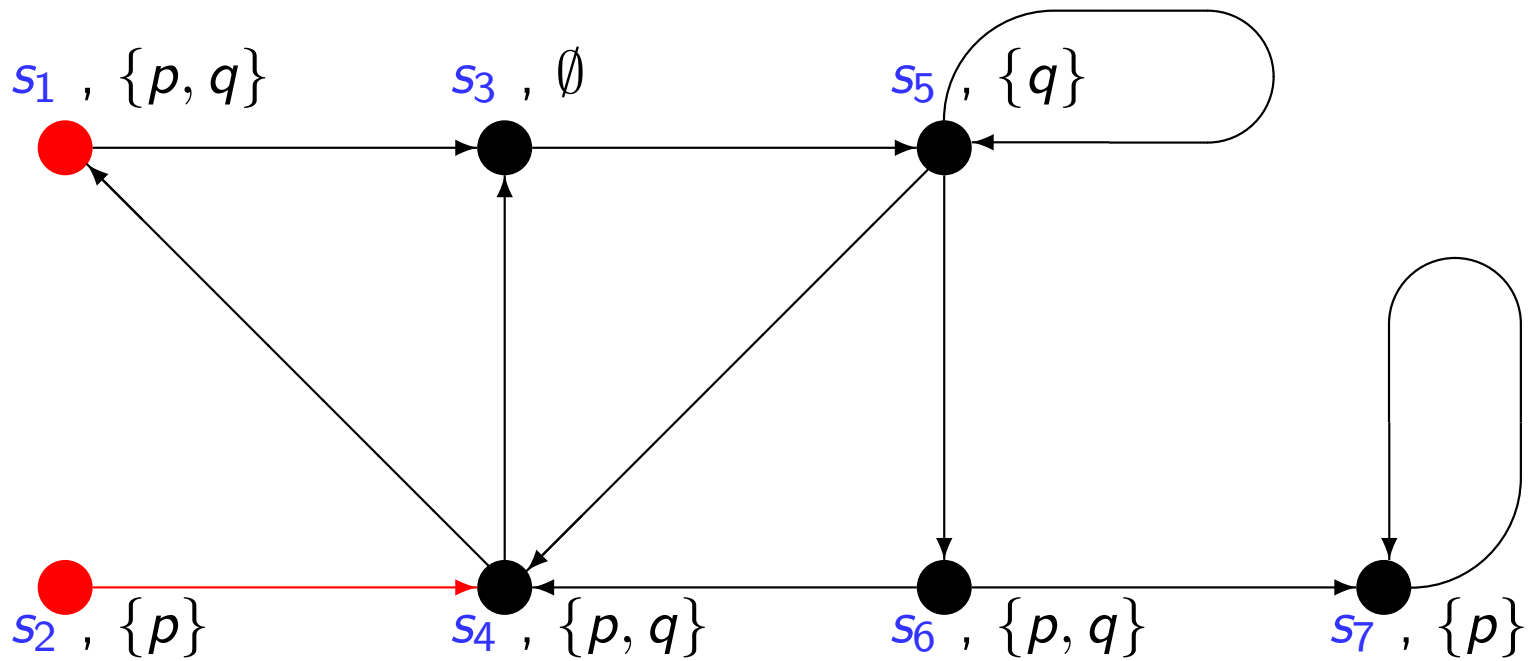
$p = \textbf{true}$
$q = \textbf{false}$



РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

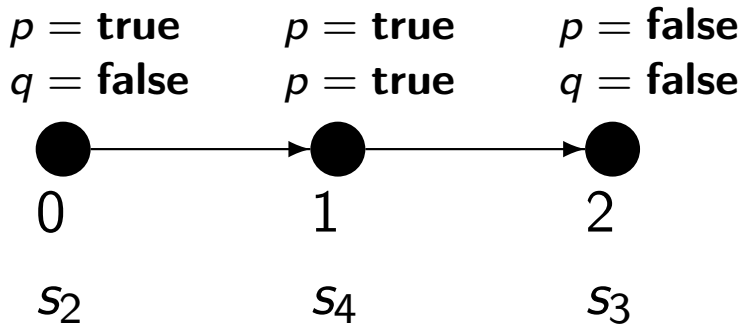
Пример LTS

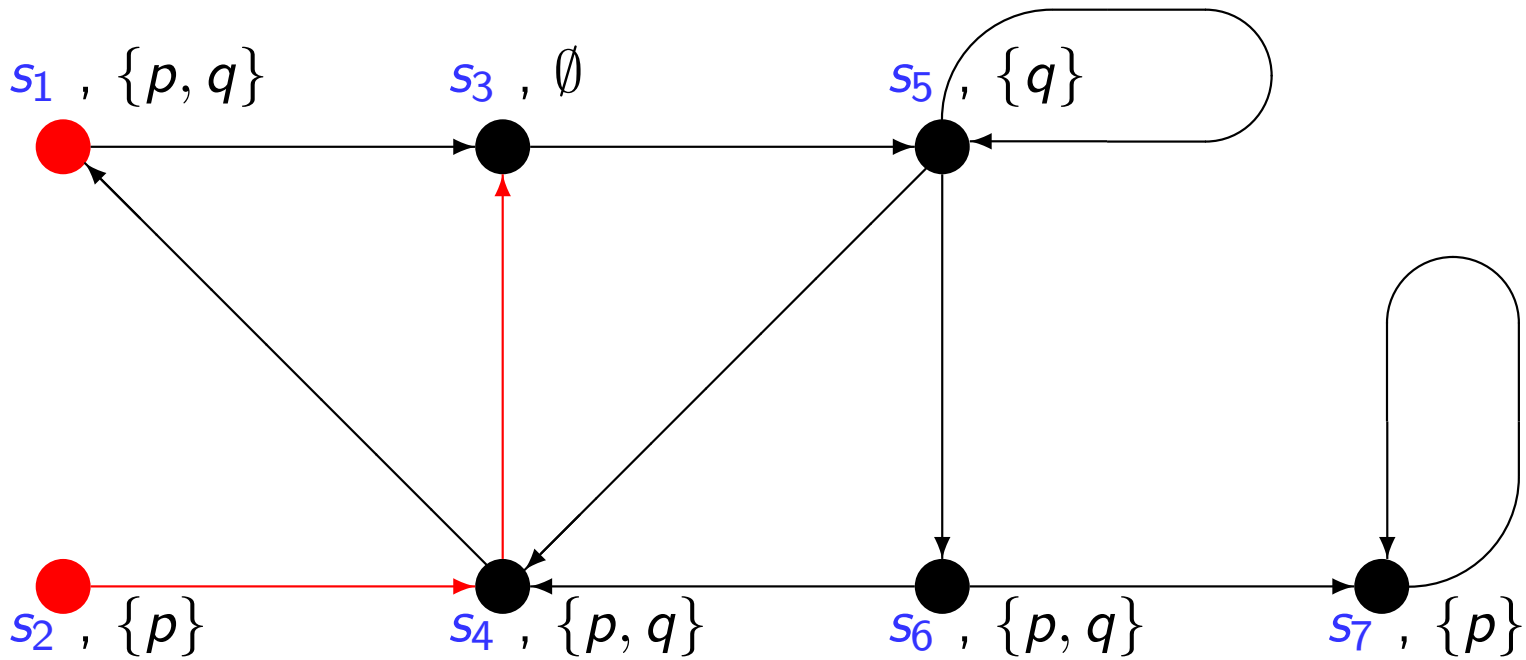




РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

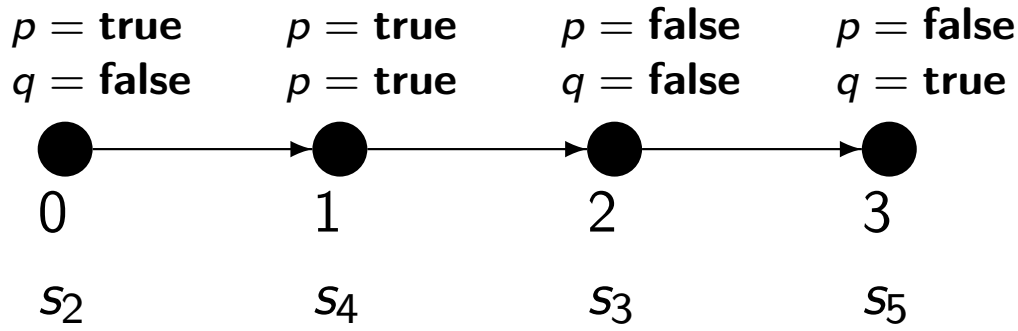
Пример LTS

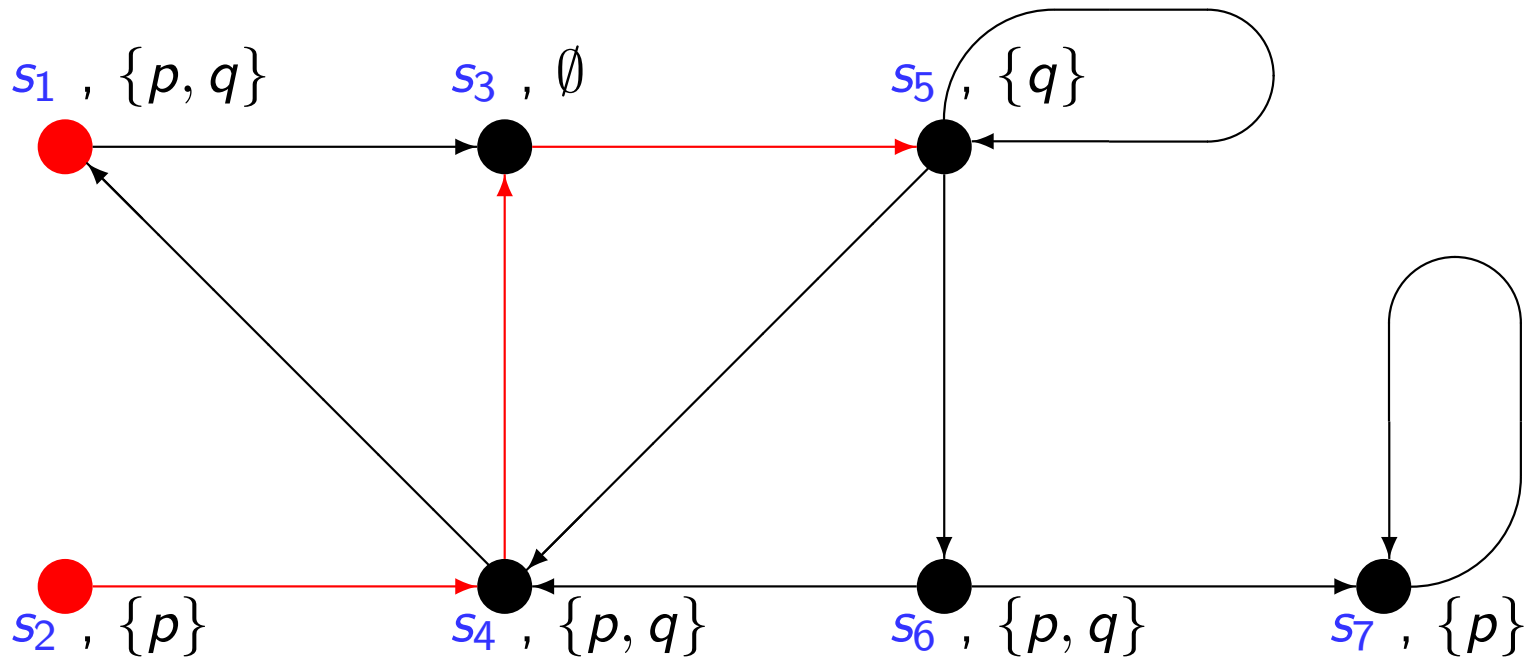




РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

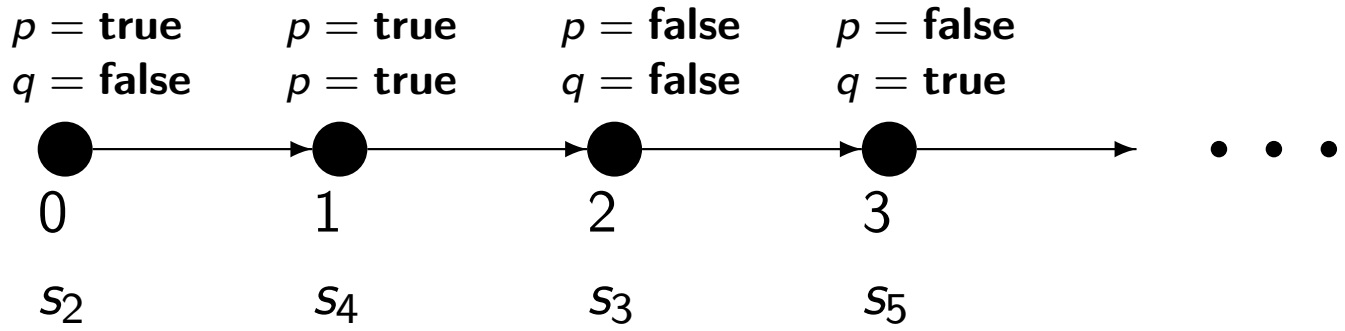
Пример LTS

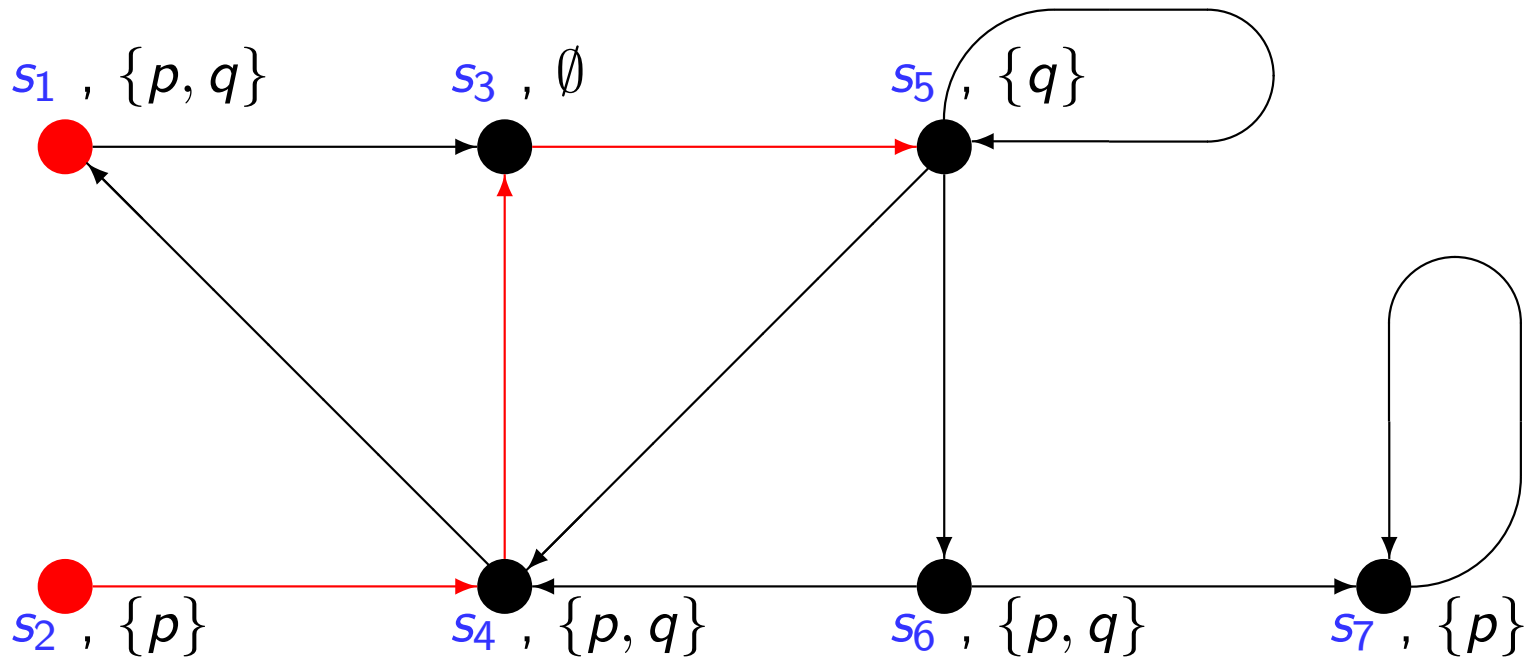




РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

Пример LTS





РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS и распределенные программы

LTS, соответствующую программе π, можно построить так:

- Состояниями LTS полагаются состояния вычисления программы. Состояние вычисления программы π — это пара (состояние управления, состояние данных). Состояние управления — это значение счетчика команд программы. Состояние данных — это подстановка, указывающая соответствие между переменными и их значениями.
- Если в точке $count_1$ программы π выполняется оператор op, преобразующий данные из состояния ξ_1 в состояние ξ_2 и передающий управление в точку $count_2$, то в LTS имеется переход $(count_1, \xi_1) \longrightarrow (count_2, \xi_2)$.
- Атомарным высказыванием p может быть любая формула логики предикатов, зависящая от переменных программы и счетчика команд. Разметка ρ определяется так: $p \in \rho((count, \xi)) \iff I_\pi \models p[(count, \xi)]$.

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

Программа драйвера π_1

while true do

L_1: **while** $R \neq$ *free* **do wait od** ;

L_2: R = *busy*;

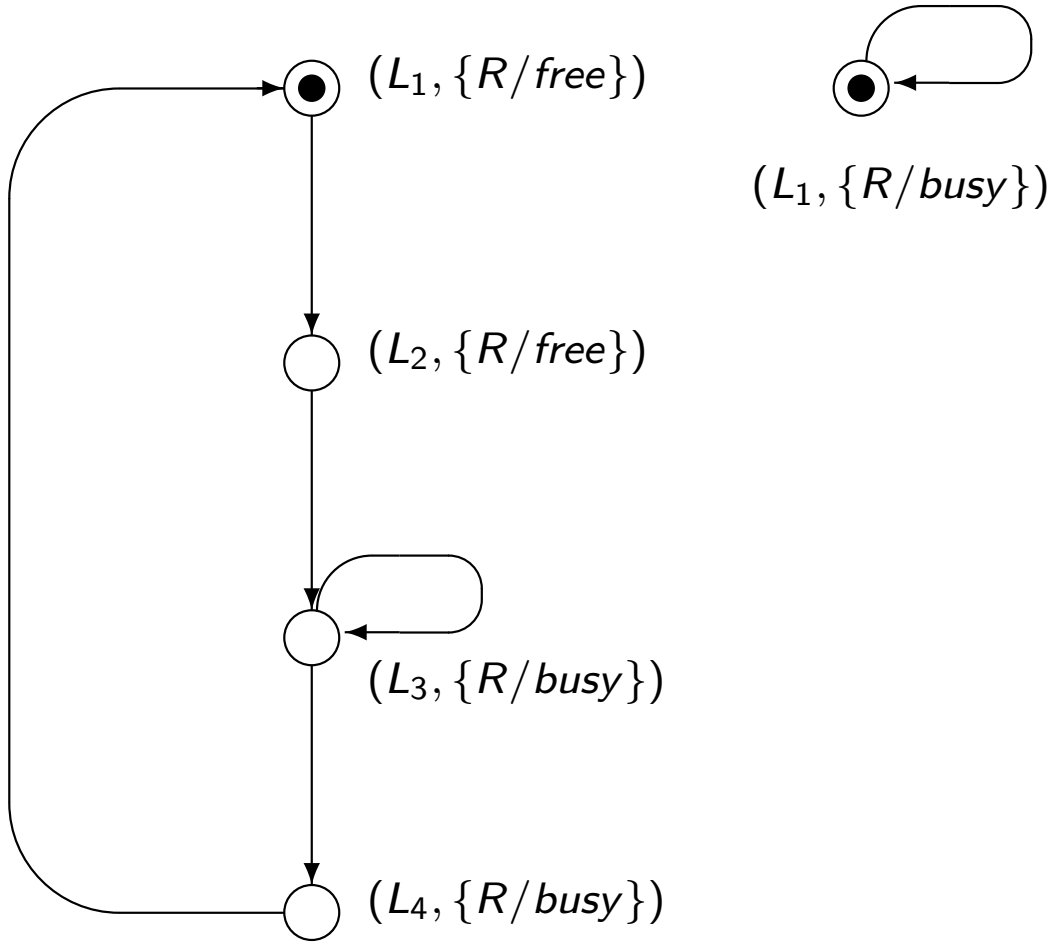
L_3: **output**(X,*printer*);

L_4: R = *free*;

od

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS для программы π_1



РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS и распределенные программы

Некоторые переменные программы могут быть доступны для других программ (т. н. разделяемые переменные) или для окружающей среды (датчики, сенсоры, средства управления).

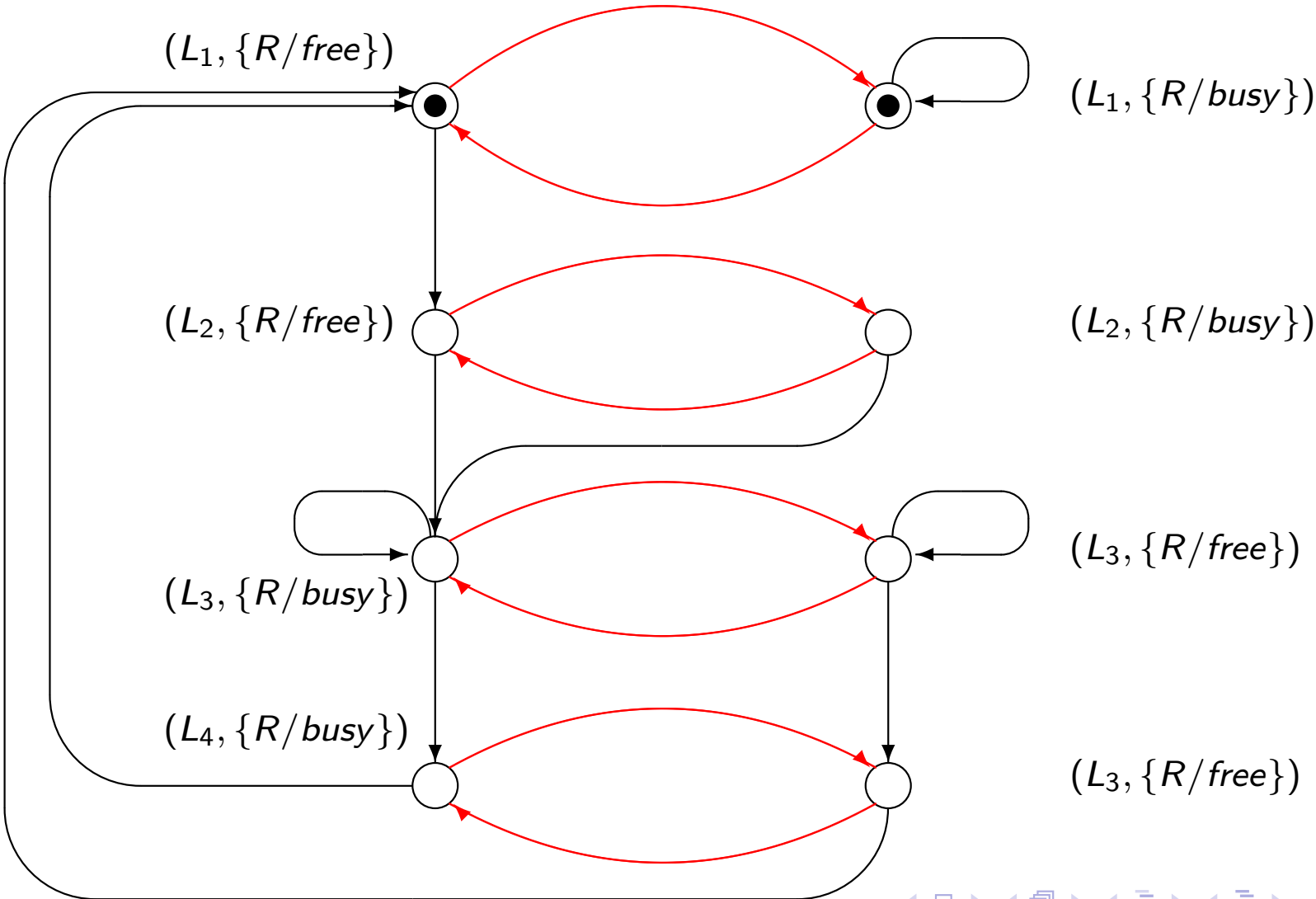
В том случае, когда программа π взаимодействует с окружающей средой, в LTS M_π вносятся следующие изменения:

для каждого состояния (l, ξ) вводятся переходы $(l, \xi) \longrightarrow (l, \xi')$ во всевозможные состояния (l, ξ'), в которых подстановки ξ', отличающиеся от ξ только значениями переменных, доступных для окружающей среды.

Например, полагая, что регистр принтера R — это тумблер, который может переключаться сколь угодно часто в разные моменты времени, получим следующую LTS, описывающую взаимодействие драйвера принтера с окружающей средой (пользователем принтера).

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS для системы $\pi_1 \parallel \mathit{environment}$



РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS и распределенные программы

LTS для распределенной системы, состоящей из двух процессов π_1 и π_2, взаимодействующих посредством разделяемых переменных, строится на основе семантики чередующихся вычислений.

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS и распределенные программы

LTS для распределенной системы, состоящей из двух процессов π_1 и π_2, взаимодействующих посредством разделяемых переменных, строится на основе семантики чередующихся вычислений.

Состояниями LTS для системы $\pi_1 \parallel \pi_2$ объявляются наборы $(count_1, count_2, \xi_1, \xi_2, \chi)$, где

- $count_1, count_2$ — значения счетчиков команд процессов π_1 и π_2,
- ξ_1, ξ_2 — подстановки, определяющие значения локальных переменных процессов π_1 и π_2,
- χ — подстановка, определяющая значения разделяемых переменных.

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS и распределенные программы

LTS для распределенной системы, состоящей из двух процессов π_1 и π_2, взаимодействующих посредством разделяемых переменных, строится на основе семантики чередующихся вычислений.

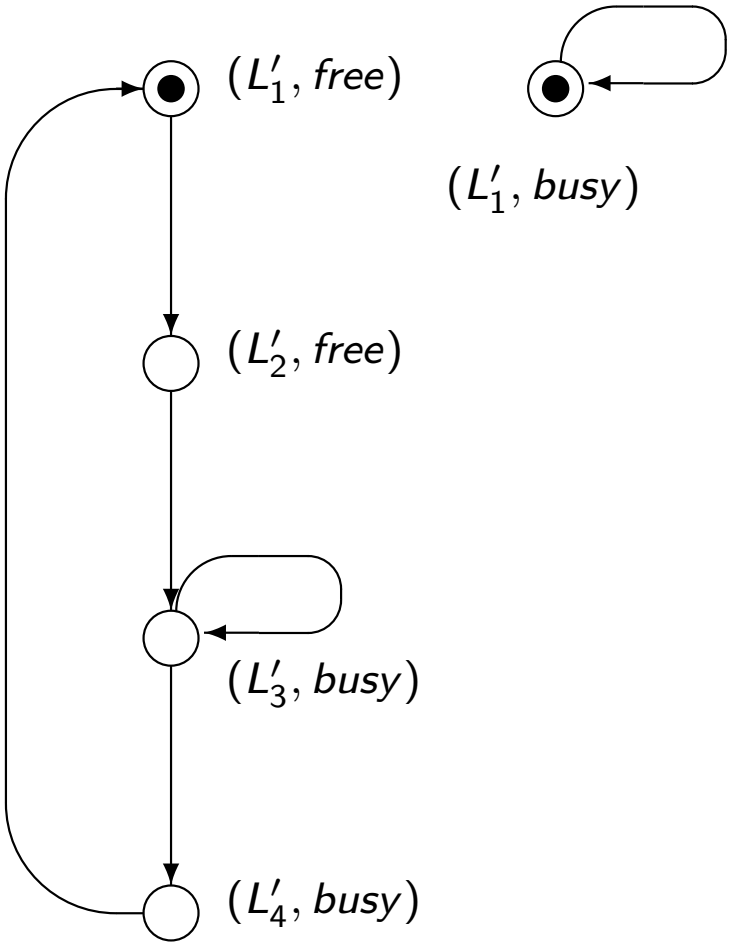
Переход $(count_1, count_2, \xi_1, \xi_2, \chi) \longrightarrow (count_1', count_2', \xi_1', \xi_2', \chi')$ возможен в том и только том случае, когда выполнено одно из двух условий:

- в LTS $M(\pi_1)$ есть переход $(count_1, \xi_1, \chi) \longrightarrow (count_1', \xi_1', \chi')$ и при этом $count_2' = count_2$, $\xi_2' = \xi_2$;
- в LTS $M(\pi_2)$ есть переход $(count_2, \xi_2, \chi) \longrightarrow (count_2', \xi_2', \chi')$ и при этом $count_1' = count_1$, $\xi_1' = \xi_1$.

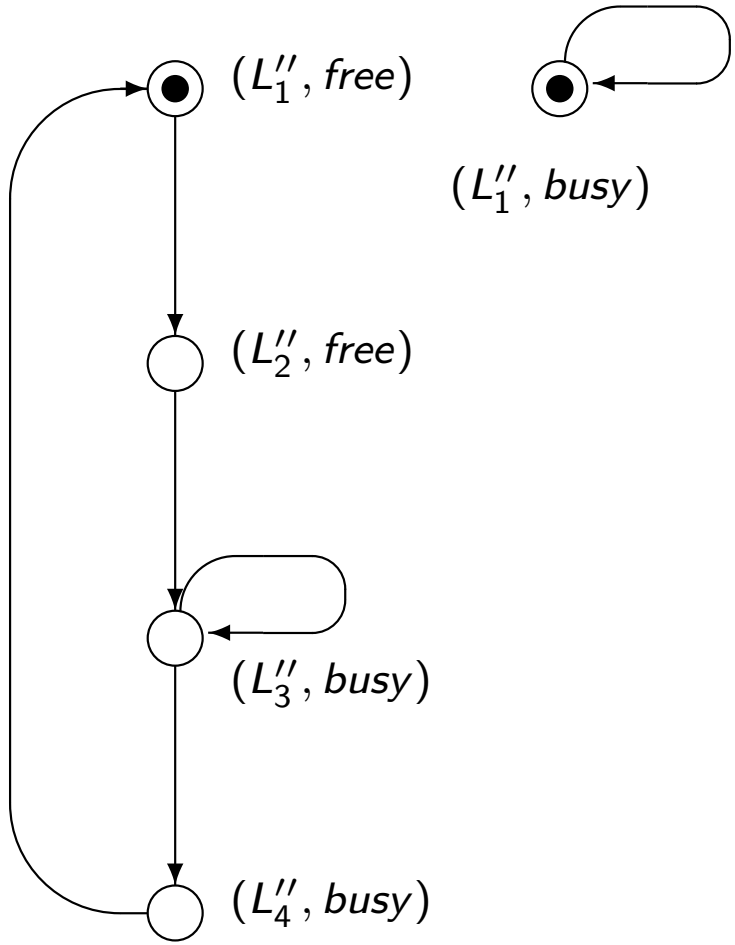
Таким образом, в семантике чередующихся вычислений параллельное выполнение процессов распределенной системы моделируется недетерминированным выбором того или иного порядка, в котором выполняются действия разных процессов.

РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS для программы π'

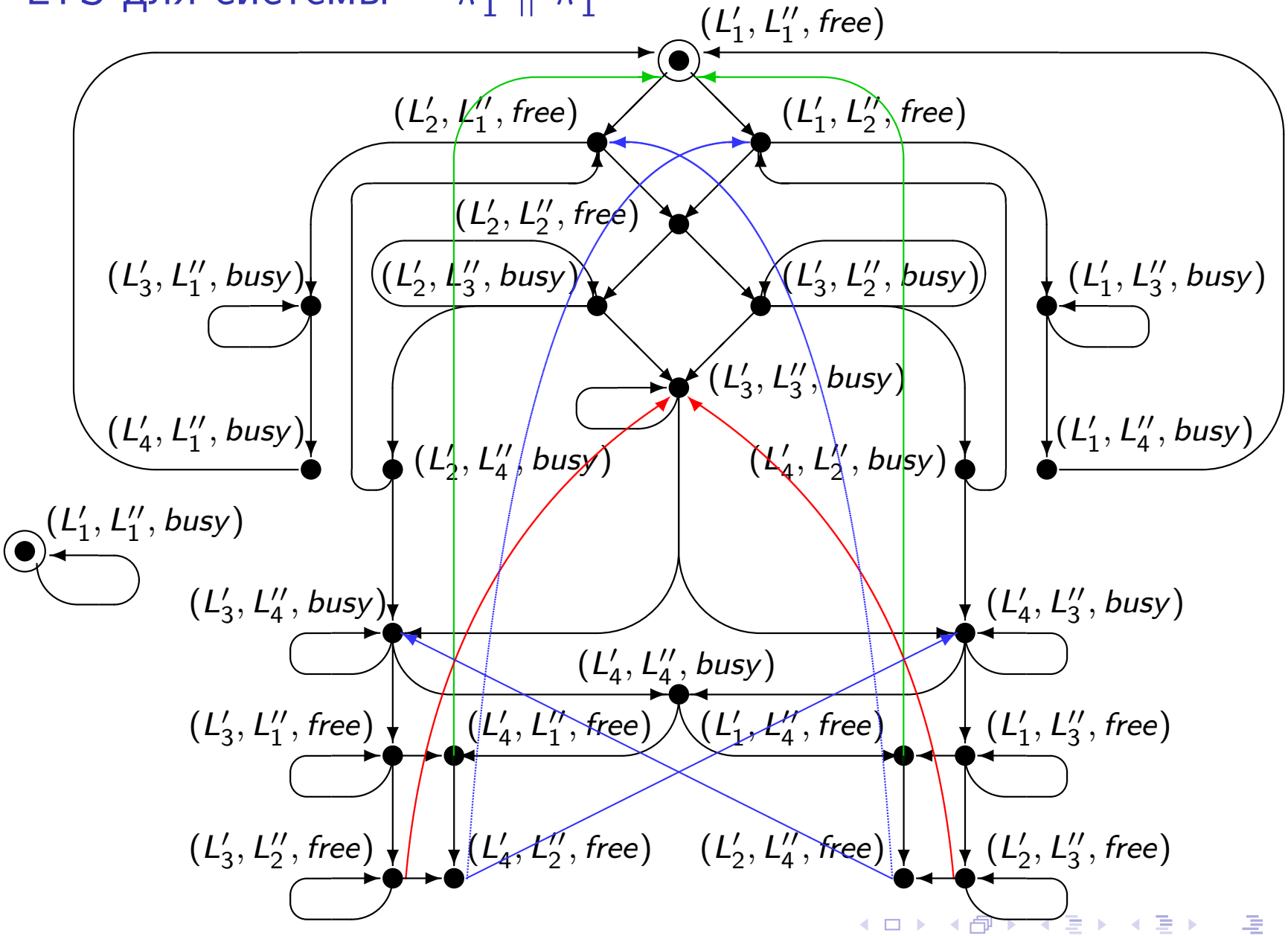


и программы π''



РАЗМЕЧЕННЫЕ СИСТЕМЫ ПЕРЕХОДОВ

LTS для системы $\pi'_1 \parallel \pi''_1$



ЗАДАЧА ВЕРИФИКАЦИИ МОДЕЛЕЙ ПРОГРАММ

Итак, для верификации распределенных программ нужны

- требования правильности (спецификации) вычислений программы, представленные формулой PLTL φ,
- математическая модель распределенной программы π, представленная конечной LTS $M(\pi)$.

Тогда для проверки правильности программы π достаточно проверить, что для любой трассы tr, $tr \in Tr_0(M(\pi))$, формула φ выполняется в темпоральной интерпретации $I(tr)$, т. е. имеет место $I(tr), 0 \models \varphi$.

Воспользуемся записью $M \models \varphi$ для обозначения утверждения «для любой трассы tr, $tr \in Tr_0(M)$, имеет место $I(tr), 0 \models \varphi$».

ЗАДАЧА ВЕРИФИКАЦИИ МОДЕЛЕЙ ПРОГРАММ

Задача верификации моделей программ (model checking) для PLTL формулируется так:

для заданной формулы PLTL φ и LTS M проверить $M \models \varphi$.

Существует ли алгоритм решения задачи верификации моделей программ?

КОНЕЦ ЛЕКЦИИ 21.