Математическая логика

(mk.cs.msu.ru → Лекционные курсы → Математическая логика (группы 318, 241))

Лекция 16

Верификация распределённых систем
Логика линейного времени (LTL)
Размеченные системы переходов
Задача model checking для LTL

Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2020, весенний семестр

Верификация распределённых систем

Рассмотрим две функции, **параллельно** и **независимо** изменяющие переменную счёт

```
void стипендия() {
  счёт += 1 000;
}
```

```
void надбавка() {
  счёт += 1 000 000;
}
```

Попробуем применить логику Хоара, чтобы проверить, корректно ли выплачиваются одна стипендия и одна надбавка согласно этим функциям

$$\{\text{счёт} = x\}$$
$$\text{счёт} := \text{счёт} + 1\,000;$$
$$\text{счёт} := \text{счёт} + 1\,000\,000$$
$$\{\text{счёт} = x + 1\,001\,000\}$$

$$\{\text{счёт} = x\}$$
$$\text{счёт} := \text{счёт} + 1\,000\,000;$$
$$\text{счёт} := \text{счёт} + 1\,000$$
$$\{\text{счёт} = x + 1\,001\,000\}$$

Ответ: **да, корректно**

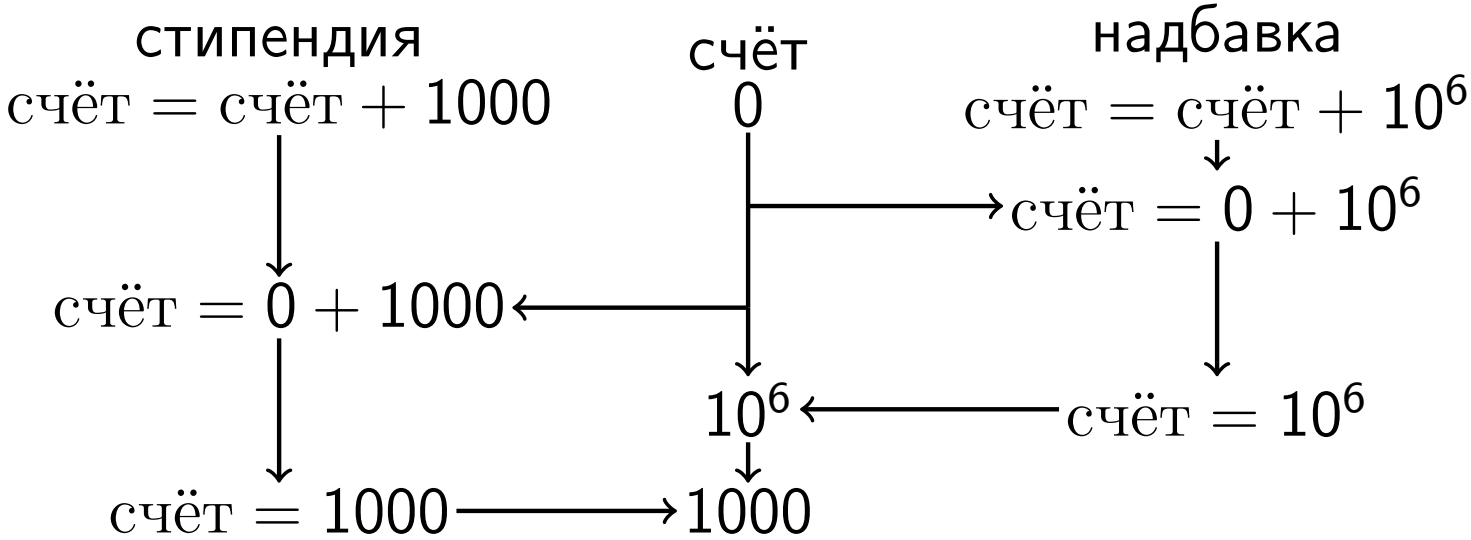
Нет ли здесь подвоха?

Верификация распределённых систем

Рассмотрим две функции, **параллельно** и **независимо** изменяющие переменную счёт

```
void стипендия() {
  счёт += 1 000;
}
```

```
void надбавка() {
  счёт += 1 000 000;
}
```



Где мой миллион?!

Верификация распределённых систем

Рассмотрим две функции, **параллельно** и **независимо** изменяющие переменную счёт

```
void стипендия() {
  счёт += 1 000;
}
```

```
void надбавка() {
  счёт += 1 000 000;
}
```

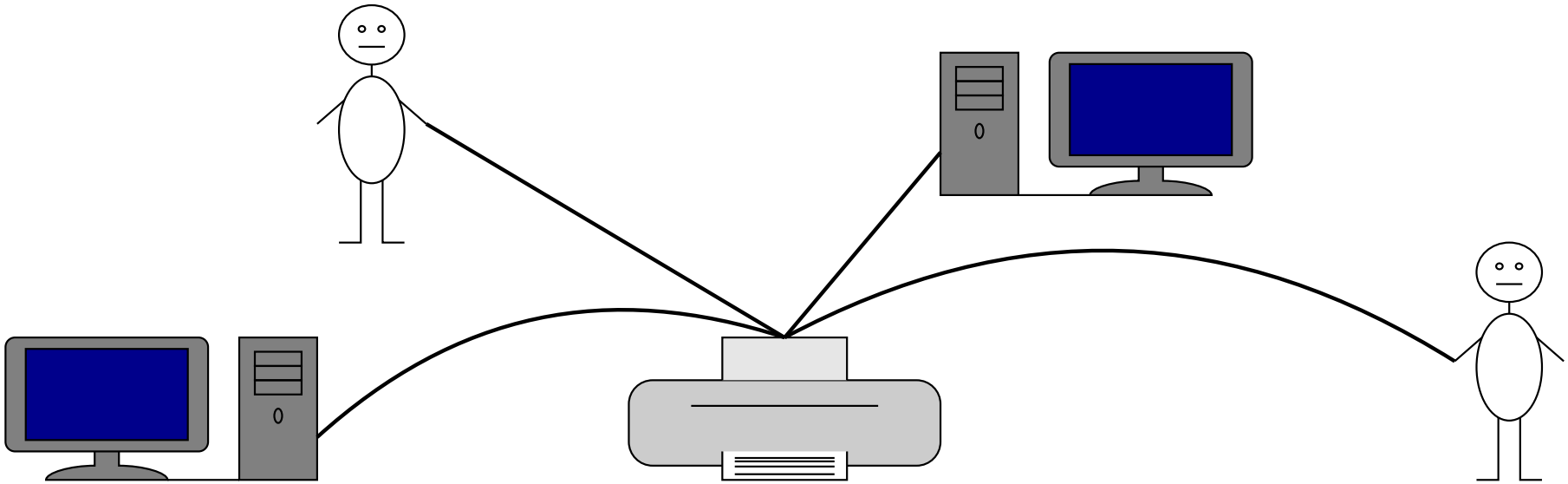
Если вызывается только одна из функций, то она всегда выполняется корректно

При этом выполнение функций, вызванных параллельно, может быть ошибочным

Ошибка проявляется крайне редко и практически не обнаруживается тестированием: чтобы она произошла, параллельно выполняющиеся функции должны произвести заданные действия *почти одновременно* — такая ситуация считается **невоспроизводимой**

Верификация распределённых систем

Ещё один пример: с сетевым принтером пытаются взаимодействовать участники *неизвестной природы*



Принтер работает последовательно: принимает информацию и производит печать согласно содержащейся в нём *программе*

Программы остальных участников, *если они есть*, неизвестны

Как могут выглядеть ошибки выполнения такой системы, и как проверить их отсутствие?

Верификация распределённых систем

Как обнаруживать такие ошибки

Предположим для простоты, что все компоненты распределённой системы — очень простые программы

Что мешает явно перебрать все сценарии выполнения программ и для каждого из них убедиться в отсутствии ошибок?

Пусть в системе параллельно работают 70 программ, каждая из которых совершает одно действие и завершается

Тогда всевозможных сценариев работы будет **70!**

это больше, чем **гугол**

При этом разные последовательности действий могут приводить к совершенно разным и неожиданным результатам

(*куда исчез мой миллион?*)

Верификация распределённых систем

Как обнаруживать такие ошибки:

- **тестирование** не подходит (*так не спасти мой миллион*)
- зато можно попробовать **формальную верификацию**

Прежде всего остановимся на том, какие **требования** обычно предъявляются к распределённым системам

В таких требованиях часто используется время:

- в тот момент, когда функции завершат работу, на счёт поступит 1 001 000
- в какой бы момент времени ни пришёл запрос на печать, когда-нибудь после этого документ обязательно напечатается

Для описания развития событий во времени предназначены *темпоральные логики* (*подробно рассмотрим только LTL*)

LTL: синтаксис формул

Далее считаем заданным

конечное множество атомарных высказываний AP

Синтаксис ltl-формул (над AP) зададим следующей БНФ:

$$\varphi ::= p \mid (\neg\varphi) \mid (\varphi \,\&\, \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\mathbf{F}\varphi) \mid (\mathbf{G}\varphi) \mid (\mathbf{X}\varphi) \mid (\varphi \mathbf{U} \varphi),$$

где φ — ltl-формула

$p \in \mathrm{AP}$

$\neg$, $\&$, $\vee$ и $\to$ — обычные логические связки

F, **G**, **X** и **U** — темпоральные модальности (операторы):

Fφ = “когда-нибудь станет верным φ”

Gφ = “φ будет верно всегда”

Xφ = “в следующий момент времени будет верно φ”

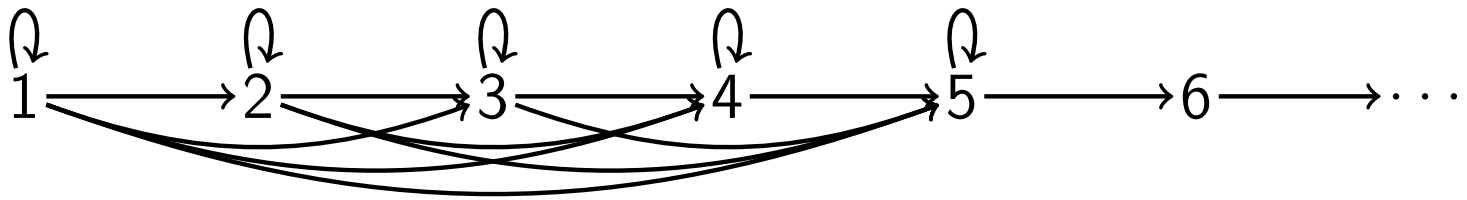
φ**U**ψ = “когда-нибудь станет верным ψ, а до тех пор всегда будет верно φ”

Приоритеты операций: F, G, X, ¬; затем U;

затем остальные связки с обычным приоритетом

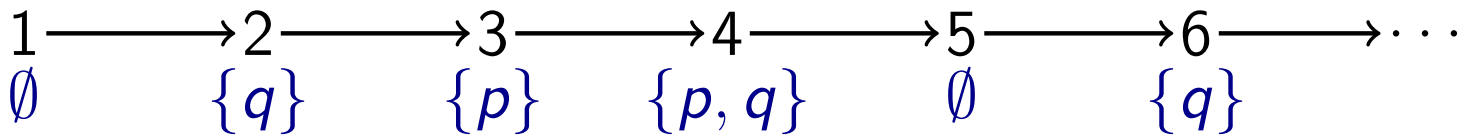
LTL: семантика формул

Напоминание: смысл формулы определяется *ltl-интерпретацией*, представляющей собой множество натуральных чисел $(1, 2, 3, \dots)$, соединённых переходами согласно порядку $(i \to j \Leftrightarrow i \leq j)$



LTL: семантика формул

Напоминание: смысл формулы определяется *ltl-интерпретацией*, представляющей собой множество натуральных чисел $(1, 2, 3, \dots)$, соединённых переходами согласно порядку $(i \to j \Leftrightarrow i \leq j)$ и размеченных множествами атомарных высказываний — например:



Более просто такая интерпретация может быть определена как **последовательность** множеств атомарных высказываний: i-м элементом последовательности $(i \geq 1)$ является оценка мира i — например, для интерпретации выше: $\emptyset, \{q\}, \{p\}, \{p, q\}, \emptyset, \{q\}, \dots$

Индексами будем называть натуральные числа: $1, 2, 3, \dots$
$S[i]$ — i-й элемент последовательности S

(*элементы нумеруются с единицы*)

LTL: семантика формул

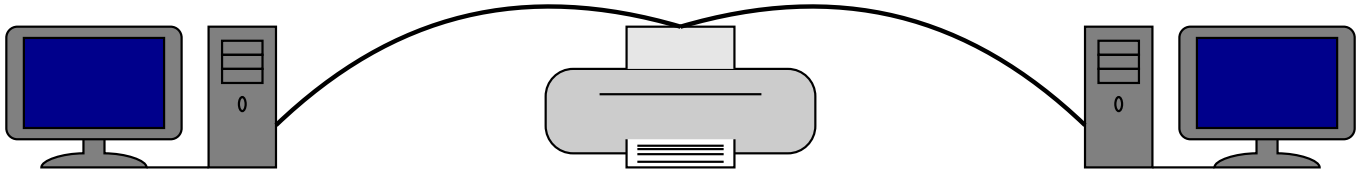
Определим семантику ltl-формул “с ноля” в новых обозначениях

Отношение выполнимости формулы в интерпретации $\mathcal{I}$ в момент времени t ($t \in \{1, 2, \ldots\}$) определяется так:

- $\mathcal{I}, t \models p \Leftrightarrow p \in \mathcal{I}[t]$, если $p \in \mathrm{AP}$
- $\mathcal{I}, t \models \neg\varphi \Leftrightarrow \mathcal{I}, t \not\models \varphi$
- $\mathcal{I}, t \models \varphi \,\&\, \psi \Leftrightarrow \mathcal{I}, t \models \varphi$ и $\mathcal{I}, t \models \psi$
- $\mathcal{I}, t \models \varphi \vee \psi \Leftrightarrow \mathcal{I}, t \models \varphi$ или $\mathcal{I}, t \models \psi$
- $\mathcal{I}, t \models \varphi \to \psi \Leftrightarrow \mathcal{I}, t \not\models \varphi$ или $\mathcal{I}, t \models \psi$
- $\mathcal{I}, t \models \mathbf{F}\varphi \Leftrightarrow$ существует индекс i, такой что $i \geq t$ и $\mathcal{I}, i \models \varphi$
- $\mathcal{I}, t \models \mathbf{G}\varphi \Leftrightarrow$
  для любого индекса i, такого что $i \geq t$, верно $\mathcal{I}, i \models \varphi$
- $\mathcal{I}, t \models \mathbf{X}\varphi \Leftrightarrow \mathcal{I}, t+1 \models \varphi$
- $\mathcal{I}, t \models \varphi \mathbf{U} \psi \Leftrightarrow$ существует индекс i, такой что: $i \geq t$; $\mathcal{I}, i \models \psi$;
  для любого индекса j, такого что $t \leq j < i$, верно $\mathcal{I}, j \models \varphi$

LTL: выразительные возможности

Вернёмся к примеру с сетевым принтером, для простоты полагая, что в сети ровно два компьютера:



Правильная работа такой системы предполагает, в числе прочего, что:

1. данные на принтер всегда передаются не более чем одним компьютером
2. компьютер не будет передавать данные на принтер вечно
3. если компьютер посылает достаточно много (*потенциально бесконечно много*) запросов на печать, то рано или поздно этот компьютер начнёт передавать данные
4. если компьютер начинает (*конечный*) сеанс передачи данных на принтер, то в течение всего сеанса принтер будет занят этим компьютером

LTL: выразительные возможности

Выделим из предложенных требований
следующие атомарные высказывания:

- rq_i: i-й компьютер посылает запрос на печать
- pr_i: i-й компьютер передаёт данные на печать
- $busy_i$: принтер занят i-м компьютером

Требования к системе с сетевым принтером можно записать в терминах LTL так:

1. данные на принтер всегда передаются не более чем одним компьютером

$$\mathbf{G}\neg(pr_1 \,\&\, pr_2)$$

LTL: выразительные возможности

2. i-й компьютер не будет передавать данные на принтер вечно

$$\neg \mathbf{FG} pr_i$$

3. если i-й компьютер посылает достаточно много запросов на печать, то рано или поздно этот компьютер начнёт передавать данные

$$\mathbf{GF} rq_i \to \mathbf{F} pr_i$$

4. если i-й компьютер начинает сеанс передачи данных на принтер, то в течение всего сеанса принтер будет занят этим компьютером

$$\mathbf{G}(pr_i \to busy_i \mathbf{U} \neg pr_i) \quad ?$$
$$\mathbf{G}(\neg pr_i \,\&\, \mathbf{X} pr_i \to \mathbf{X}(busy_i \mathbf{U} \neg pr_i)) \quad ?$$

Законы логики линейного времени

Пусть $\mathcal{I}$ — интерпретация, и φ, ψ — ltl-формулы

Тогда

- формула φ выполняется в интерпретации $\mathcal{I}$ ($\mathcal{I} \models \varphi$), если верно $\mathcal{I}, 1 \models \varphi$
- формула φ общезначима ($\models \varphi$), если для любой интерпретации $\mathcal{J}$ верно $\mathcal{J} \models \varphi$
- формулы φ, ψ равносильны ($\varphi \sim \psi$), если верно $\models \varphi \leftrightarrow \psi$

В логике линейного времени выполняются
все законы модальных логик

К этим законам добавляются и другие,
позволяющие преобразовывать подформулы,
содержащие темпоральные операторы

Законы логики линейного времени

Законы отрицания

$\neg \mathbf{X}\varphi \sim \mathbf{X}\neg\varphi$

$\neg \mathbf{F}\varphi \sim \mathbf{G}\neg\varphi$

$\neg \mathbf{G}\varphi \sim \mathbf{F}\neg\varphi$

$\neg(\varphi \mathbf{U} \psi) \sim \mathbf{G}\neg\psi \vee (\neg\psi \mathbf{U}(\neg\varphi \,\&\, \neg\psi))$

Законы исключения

$\mathbf{F}\varphi \sim \neg\mathbf{G}\neg\varphi$

$\mathbf{G}\varphi \sim \neg\mathbf{F}\neg\varphi$

$\mathbf{F}\varphi \sim \mathbf{true}\, \mathbf{U}\varphi$

Законы неподвижной точки

$\mathbf{F}\varphi \sim \varphi \vee \mathbf{X}\mathbf{F}\varphi$

$\mathbf{G}\varphi \sim \varphi \,\&\, \mathbf{X}\mathbf{G}\varphi$

$\varphi \mathbf{U} \psi \sim \psi \vee (\varphi \,\&\, \mathbf{X}(\varphi \mathbf{U} \psi))$

Доказательство. Достаточно вспомнить определение выполнимости

Подробно докажем справедливость закона

$$\neg(\varphi \mathbf{U} \psi) \sim \mathbf{G}\neg\psi \vee (\neg\psi \mathbf{U}(\neg\varphi \,\&\, \neg\psi))$$

Законы логики линейного времени

Доказательство. $\neg(\varphi \mathbf{U} \psi) \sim \mathbf{G}\neg\psi \vee (\neg\psi \mathbf{U}(\neg\varphi \,\&\, \neg\psi))$

Рассмотрим произвольную интерпретацию $\mathcal{I}$

$\mathcal{I} \models \neg(\varphi \mathbf{U} \psi) \Leftrightarrow$ (*семантика* $\neg$)

$\mathcal{I} \not\models (\varphi \mathbf{U} \psi) \Leftrightarrow$ (*семантика* **U**)

верно хотя бы одно из двух:

- ни для какого индекса t не верно $\mathcal{I}, t \models \psi$
  - что равносильно $\mathcal{I} \models \mathbf{G}\neg\psi$ (по *семантике* $\neg$ *и* **G**)
- существует индекс t, такой что $\mathcal{I}, t \not\models \varphi$
  и для любого индекса i, такого что $1 \leq i \leq t$, верно $\mathcal{I}, i \not\models \psi$
  - то есть существует момент t, такой что $\mathcal{I}, t \models \neg\varphi \,\&\, \neg\psi$ и для любого момента j, такого что $1 \leq j < t$, верно $\mathcal{I}, j \models \neg\psi$
  - что равносильно $\mathcal{I} \models \neg\psi \mathbf{U}(\neg\varphi \,\&\, \neg\psi)$ (по *семантике* **U**)

$\Leftrightarrow$ (*семантика* $\vee$)

$\mathcal{I} \models \mathbf{G}\neg\psi \vee \neg\psi \mathbf{U}(\neg\varphi \,\&\, \neg\psi)$

Законы логики линейного времени

Задача для самостоятельного размышления

(законы дистрибутивности)

- $\mathbf{F}(\varphi \vee \psi) \stackrel{?}{\sim} \mathbf{F}\varphi \vee \mathbf{F}\psi$
- $\mathbf{F}(\varphi \,\&\, \psi) \stackrel{?}{\sim} \mathbf{F}\varphi \,\&\, \mathbf{F}\psi$
- $\mathbf{G}(\varphi \vee \psi) \stackrel{?}{\sim} \mathbf{G}\varphi \vee \mathbf{G}\psi$
- $\mathbf{G}(\varphi \,\&\, \psi) \stackrel{?}{\sim} \mathbf{G}\varphi \,\&\, \mathbf{G}\psi$
- $\varphi \mathbf{U} (\psi \vee \chi) \stackrel{?}{\sim} (\varphi \mathbf{U} \psi) \vee (\varphi \mathbf{U} \chi)$
- $\varphi \mathbf{U} (\psi \,\&\, \chi) \stackrel{?}{\sim} (\varphi \mathbf{U} \psi) \,\&\, (\varphi \mathbf{U} \chi)$
- $(\varphi \vee \psi) \mathbf{U} \chi \stackrel{?}{\sim} (\varphi \mathbf{U} \chi) \vee (\psi \mathbf{U} \chi)$
- $(\varphi \,\&\, \psi) \mathbf{U} \chi \stackrel{?}{\sim} (\varphi \mathbf{U} \chi) \vee (\psi \mathbf{U} \chi)$

Размеченные системы переходов

Чтобы было возможно применить формальную верификацию, помимо языка записи требований необходимо иметь математическую модель системы, правильность выполнения которой оценивается

При использовании LTL для записи требований чаще всего в качестве такой модели используется особенный граф:
размеченная система переходов (Labelled Transition System; LTS)

Рассматриваемый далее вид LTS имеет более точное название “модель Крипке”, но чтобы не путать системы переходов с интерпретациями модальных логик, будем использовать общее название “система переходов”

Размеченные системы переходов

Система переходов (с.п.) над множеством AP — это система $(S, S_0, \to, \rho)$, где:

- S — произвольное непустое конечное множество состояний
- S_0 — множество начальных состояний, $S_0 \subseteq S$
- $\to \subseteq S \times S$ — **тотальное** отношение переходов
- $\rho : S \to 2^{AP}$ — функция разметки состояний истинными в них атомарными высказываниями

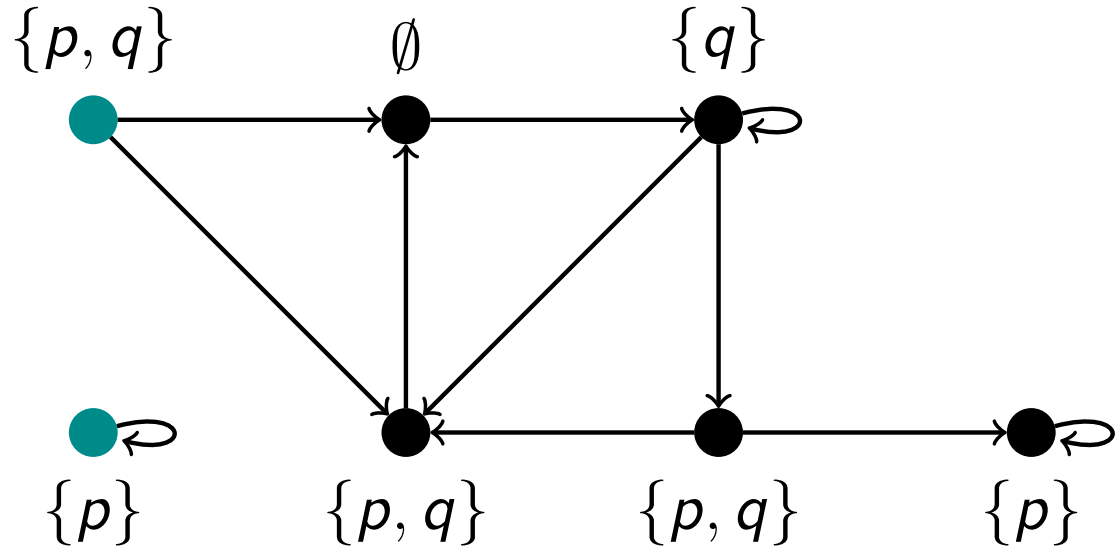
Тотальность отношения переходов означает, что из любого состояния вычисления s можно совершить переход:
для любого состояния s существует состояние s', такое что $s \to s'$

Размеченные системы переходов

Пример

$\mathrm{AP} = \{p, q\}$

Система переходов $(\{\ldots, \bullet, \ldots\}, \{\ldots, \bullet, \ldots\}, \to, \rho)$:



Размеченные системы переходов

Трасса системы переходов $M = (S, S_0, \to, \rho)$ — это любая бесконечная последовательность состояний τ вида

$$s_1 \to s_2 \to s_3 \to \dots$$

Трасса τ называется начальной, если $s_1 \in S_0$

Начальная трасса также называется вычислением с.п.

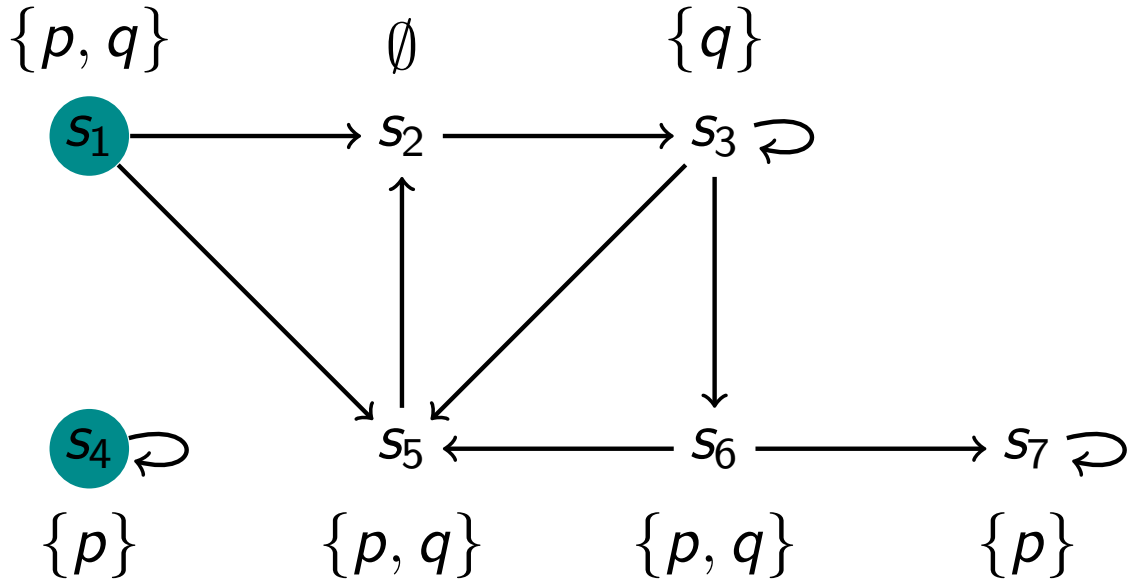
$\mathcal{I}_M(\tau)$ — интерпретация, порождаемая трассой τ с.п. M:

$$\rho(s_1), \rho(s_2), \rho(s_3), \dots$$

Размеченные системы переходов

Пример

Система переходов M:



Вычисление τ с.п. M: $s_1 \to s_5 \to s_2 \to s_3 \to s_3 \to \ldots$

Интерпретация $\mathcal{I}_M(\tau)$: $\{p,q\}\ ,\ \{p,q\}\ ,\ \emptyset\ ,\ \{q\}\ ,\ \{q\}\ ,\ \ldots$

Размеченные системы переходов

Как построить систему переходов, описывающую поведение распределённой системы

Начнём с простого: покажем, как можно построить с.п. для последовательной программы π

- состояния с.п. — это состояния вычисления программы π
- начальные состояния с.п. — это всевозможные состояния, с которых может начинаться вычисление π
- переход в с.п. соответствует шагу вычисления программы π
- атомарное высказывание соответствует заранее заданному свойству состояний вычисления программы
- высказывание включается в метку состояния с.п. $\Leftrightarrow$ состояние вычисления обладает соответствующим свойством

Размеченные системы переходов

Пример

Вернёмся ещё раз к примеру с сетевым принтером

Предположим, что в контроллере принтера есть однобитовый регистр **R**, доступный на чтение и запись всем компьютерам в сети:

$$\mathbf{R} = \mathbf{true} \quad \Leftrightarrow \quad \text{принтер свободен}$$

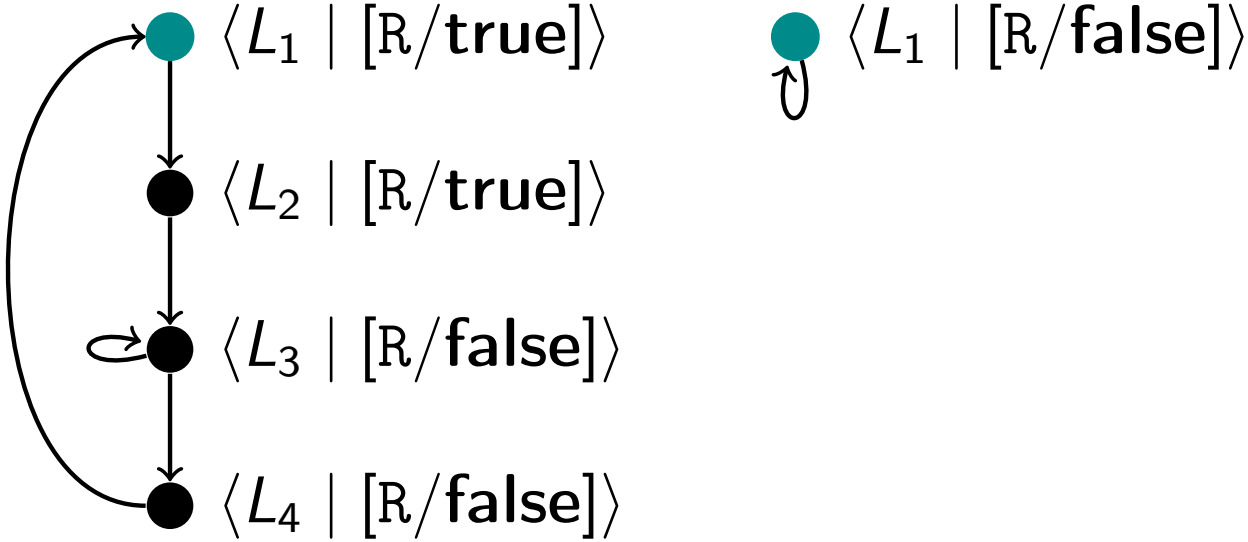
Тогда программа π взаимодействия компьютера с принтером может выглядеть так:

$$\begin{array}{l} \mathbf{while}\ (\mathbf{true})\ \{ \\ \quad L_1 : \mathbf{while}\ (!R); \\ \quad L_2 : R = \mathbf{false}; \\ \quad L_3 : SEND\_DATA \\ \quad L_4 : R = \mathbf{true}; \\ \} \end{array}$$

Размеченные системы переходов

Пример: система переходов для π может выглядеть так:

(*функция разметки опущена*)



Размеченные системы переходов

Системы переходов и окружение программ

Программа в распределённой системе может взаимодействовать с другими программами: общие переменные, обмен сообщениями, сигналы, ...

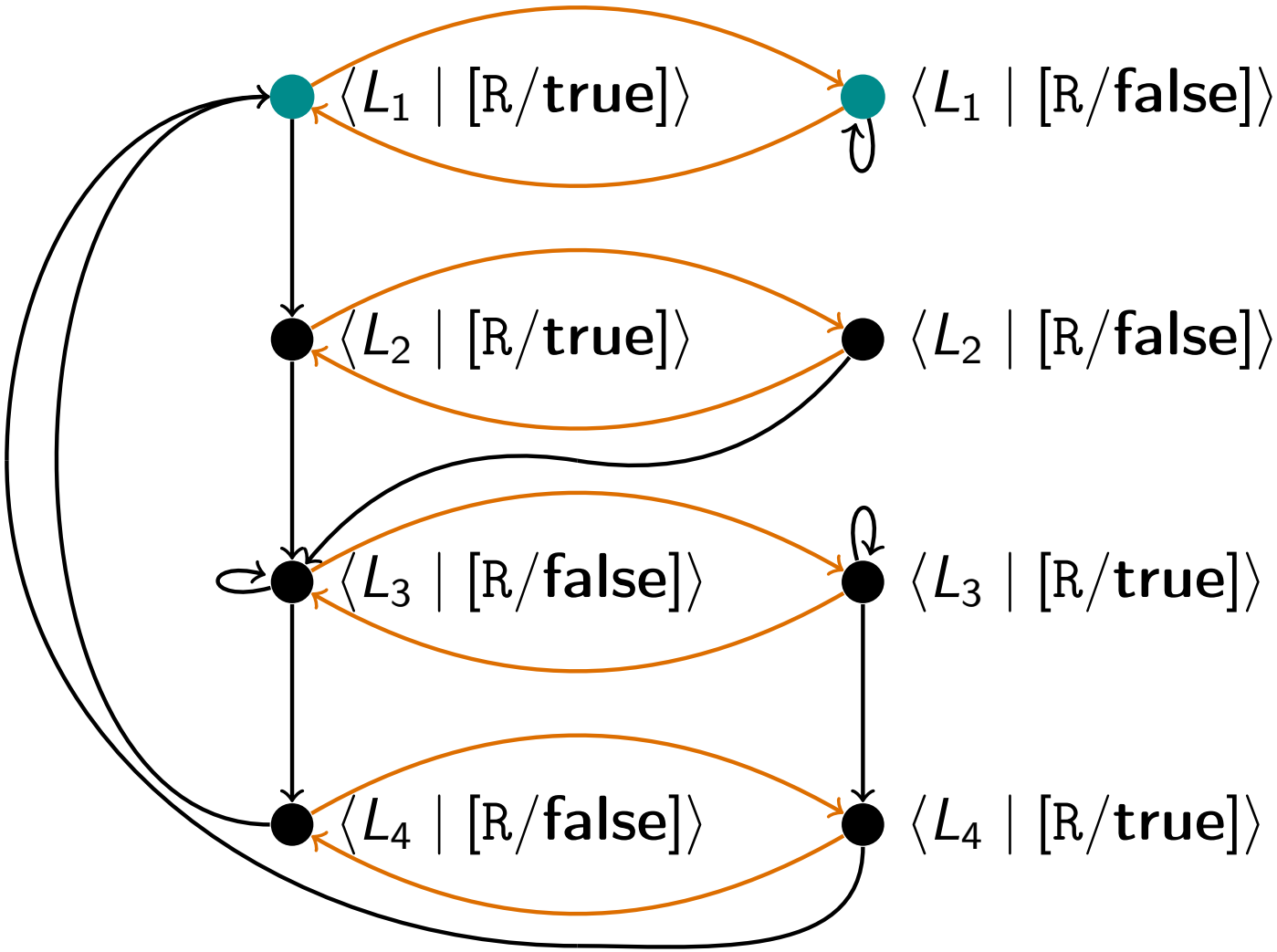
Такое взаимодействие выражается в том, что состояние вычисления программы может измениться под воздействием её **окружения**

Например, регистр **R** в последнем примере может быть изменён любым компьютером сети

Чтобы учесть такое изменение, следует добавить в с.п. переходы, отвечающие всем возможностям окружения повлиять на состояние вычисления программы

Размеченные системы переходов

Пример: система переходов для программы π с окружением, способным произвольно переключать значение регистра **R**



Размеченные системы переходов

Системы переходов и взаимодействие программ

Рассмотрим программы π_1, π_2, выполняющиеся параллельно и взаимодействующие между собой

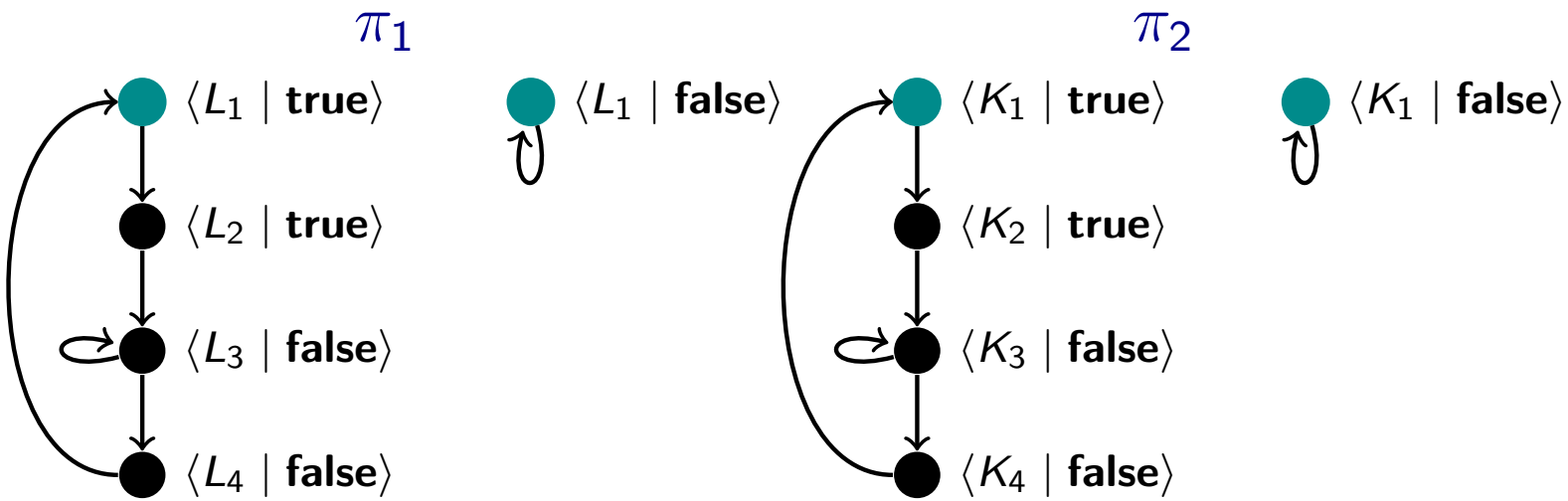
Один из наиболее популярных способов организации такого взаимодействия — это семантика чередующихся вычислений:
(*interleaving semantics*)

- совокупное состояние вычисления — это состояния π_1, π_2, в которых общие переменные изменяются синхронно (и учтены другие взаимосвязи состояний, если они есть)
- переход в с.п., описывающей параллельное выполнение программ, выглядит так:
  - произвольно (недетерминированно) выбирается одна из программ
  - выполнение перехода в с.п. = выполнение одного шага вычисления выбранной программы

Размеченные системы переходов

Пример

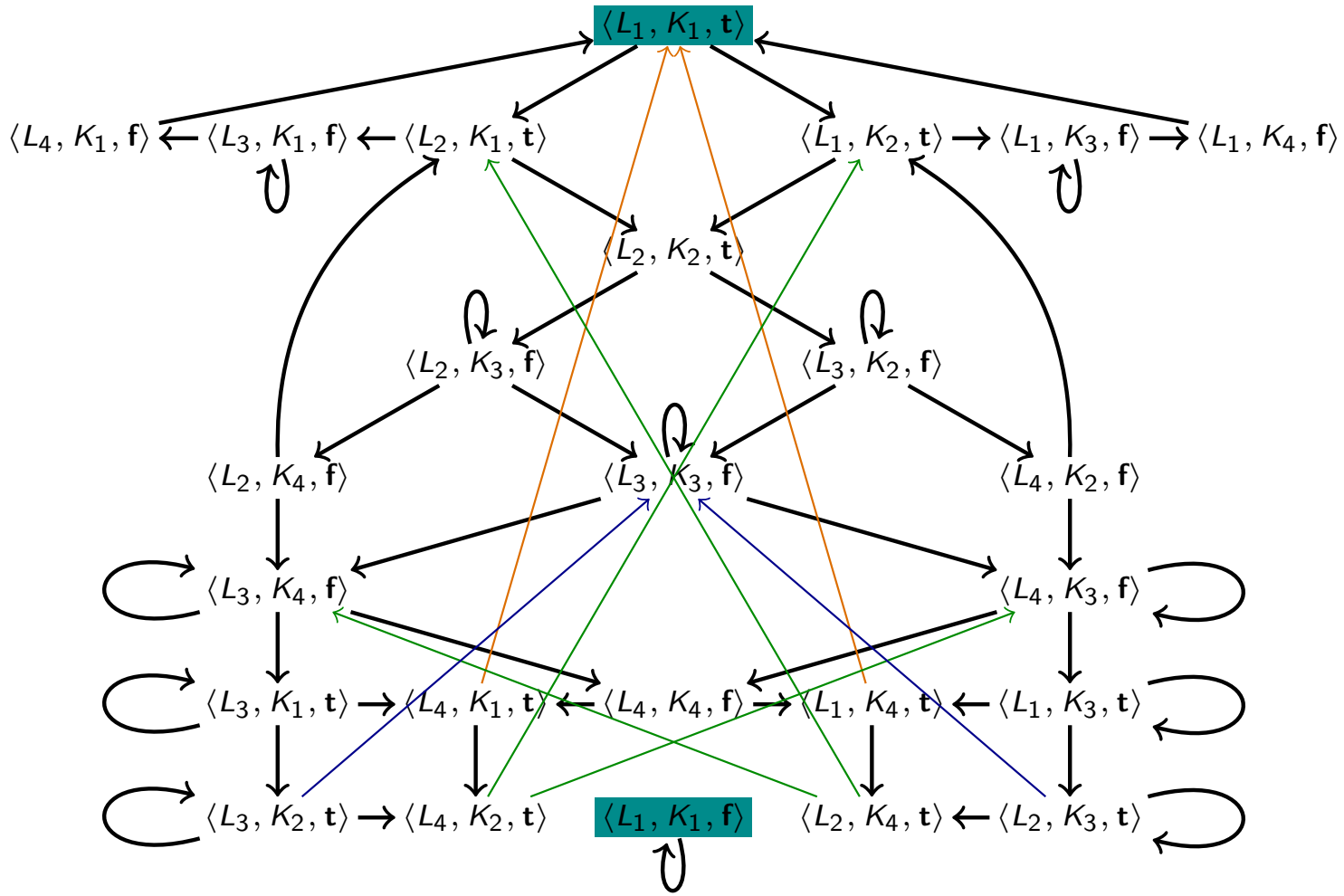
Рассмотрим две (одинаковые) программы взаимодействия с сетевым принтером, выполняющиеся согласно следующим системам переходов:



Считаем регистр **R общей переменной** программ π_1, π_2

Размеченные системы переходов

Пример: система переходов, описывающая взаимодействие π_1, π_2 согласно **семантике чередующихся вычислений**, выглядит так:



Задача model checking для LTL

$\mathfrak{I}(M)$ — множество всевозможных интерпретаций $\mathcal{I}_M(\tau)$, где τ — вычисление M

Ltl-формула φ над множеством атомарных высказываний AP выполняется на системе переходов M над тем же множеством AP ($M \models \varphi$), если для любой интерпретации $\mathcal{I}$ множества $\mathfrak{I}(M)$ справедливо соотношение $\mathcal{I} \models \varphi$

Задача верификации моделей (model checking) для LTL формулируется так:

для заданных ltl-формулы φ и системы переходов M проверить справедливость соотношения $M \models \varphi$