Математические методы верификации схем и программ

Лекторы:

Захаров Владимир Анатольевич

Подымов Владислав Васильевич

e-mail рассказчика:

valdus@yandex.ru

Осень 2016

Лекция 9

Системы реального времени

Временные автоматы

Неправдоподобные вычисления

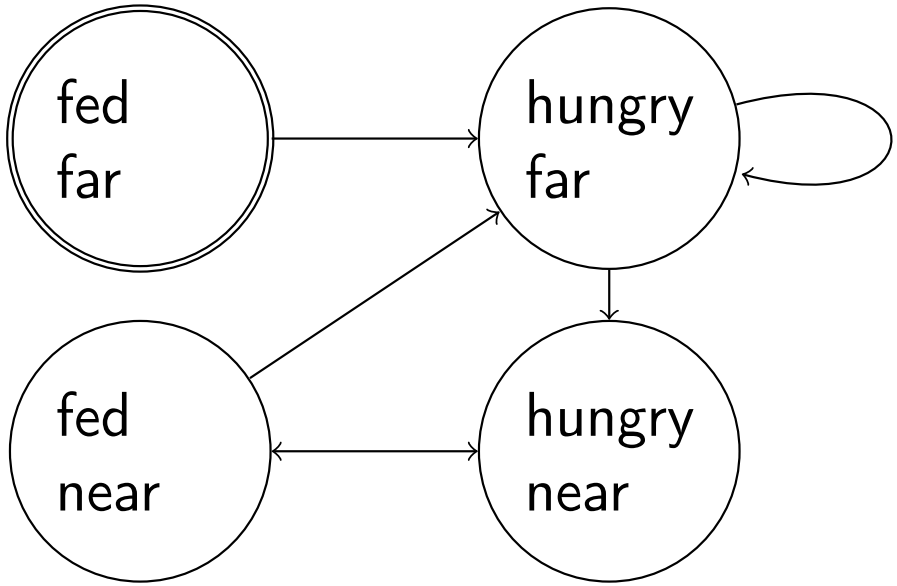
Сети временных автоматов

Timed CTL

Вступление

Пример

(*птица и рой комаров, как было много раз на семинарах*)



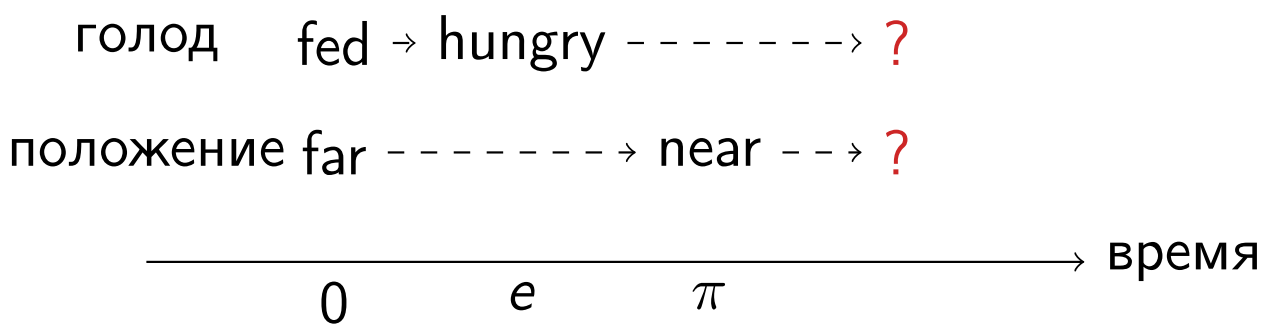
Эта модель Крипке содержит, например, такую трассу:

$$\begin{matrix} \text{fed} \\ \text{far} \end{matrix} \longrightarrow \begin{matrix} \text{hungry} \\ \text{far} \end{matrix} \longrightarrow \begin{matrix} \text{hungry} \\ \text{near} \end{matrix} \longrightarrow \begin{matrix} \text{fed} \\ \text{near} \end{matrix} \longrightarrow \begin{matrix} \cdots \\ \cdots \end{matrix}$$

Недостаток такой модели (*возможно*) в том, что и птица, и рой комаров живут в **реальном** времени, а в трассе отражено развитие системы в **дискретном** времени

Вступление

При описании системы с учётом реального времени изменение состояний компонентов системы происходит в *какие-то* (не обязательно целочисленные) моменты времени



Успеет ли птица съесть комара из роя?

Или же рой комаров успеет заметить птицу и улететь?

Это зависит от того, какими **временными** свойствами обладают компоненты системы:

- если птица успеет схватить комара до поднятия роем тревоги, то исход будет один
- если рой комаров оказывается быстрее птицы, то исход будет другой

Системы реального времени

Система реального времени (СРВ) — это система, поведение которой существенно зависит не только от того, в каком порядке изменяются состояния компонентов системы, но и от того, **за какое время** происходит изменение состояний

Для верификации СРВ необходимо иметь

1. средство описания моделей, учитывающее поведение модели в реальном времени
2. формальный язык описания требований, учитывающий реальное время
3. алгоритм проверки требований для составленной модели

Системы реального времени

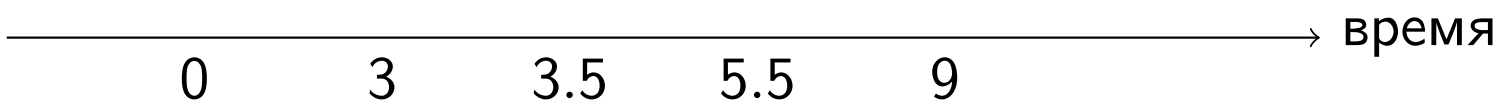
Пример системы, учитывающей временные характеристики

- птица переваривает комара ровно 3 минуты
- поголодав 5 минут, птица умирает
- голодная птица не более чем за 1 минуту подлетает к рою комаров
- птице требуется не более 2 минут, чтобы поймать комара, когда она подлетела к рою
- рой замечает птицу не менее чем за 3 минуты
- заметив птицу, рой немедленно улетает

Тогда развитие системы птица-рой во времени может выглядеть так:

голод fed → hungry - - - - - - → fed - - - - - - - → ···

положение far - - - - - - - → near - - - - - - - → far - - → ···



Системы реального времени

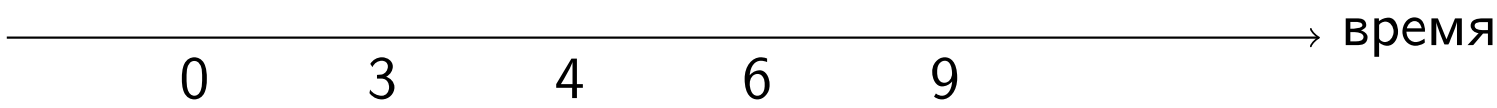
Пример системы, учитывающей временные характеристики

- птица переваривает комара ровно 3 минуты
- поголодав 5 минут, птица умирает
- голодная птица не более чем за 1 минуту подлетает к рою комаров
- птице требуется не более 2 минут, чтобы поймать комара, когда она подлетела к рою
- рой замечает птицу не менее чем за 3 минуты
- заметив птицу, рой немедленно улетает

Тогда развитие системы птица-рой во времени может выглядеть так:

голод fed → hungry - - - - - - → fed - - - - - - - → ···

положение far - - - - - - - → near - - - - - - - → far - - → ···

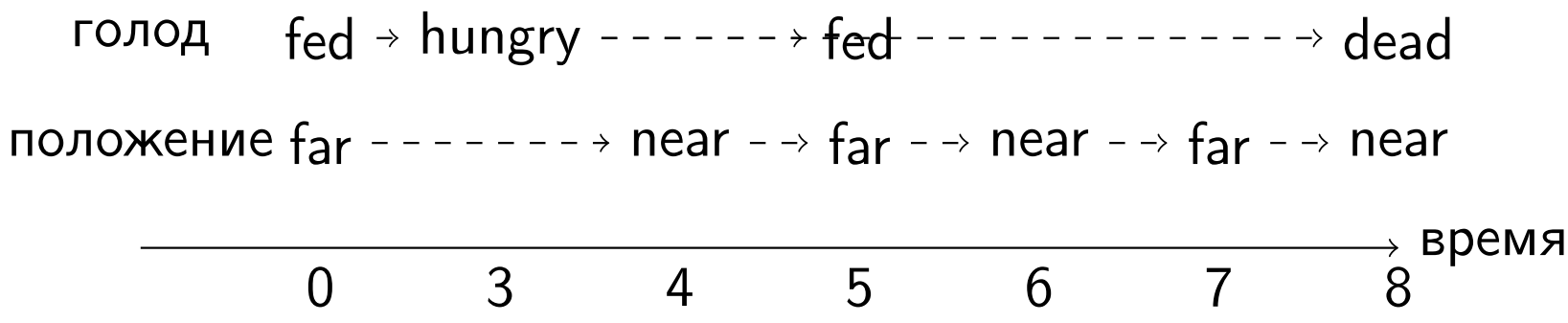


Системы реального времени

Пример системы, учитывающей временные характеристики

- птица переваривает комара ровно 3 минуты
- поголодав 5 минут, птица умирает
- голодная птица не более чем за 1 минуту подлетает к рою комаров
- птице требуется не более 2 минут, чтобы поймать комара, когда она подлетела к рою
- рой замечает птицу не менее чем за 1 минуту
- заметив птицу, рой немедленно улетает

Тогда развитие системы птица-рой во времени может выглядеть так:



Системы реального времени

Пример системы, учитывающей временные характеристики

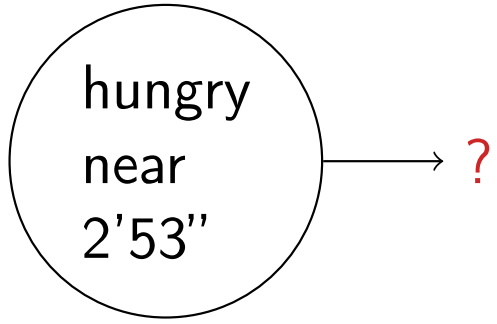
- птица переваривает комара ровно 3 минуты
- поголодав 5 минут, птица умирает
- голодная птица не более чем за 1 минуту подлетает к рою комаров
- птице требуется не более 2 минут, чтобы поймать комара, когда она подлетела к рою
- рой замечает птицу не менее чем за 1 минуту
- заметив птицу, рой немедленно улетает

В зависимости от того, насколько быстро будут действовать птица и рой, птица может как накормиться, так и умереть

Системы реального времени

И как же учесть реальное время в модели Крипке?

Давайте попробуем явно добавить время в состояние:



Реальное время континуально, а число элементов трассы счётно, поэтому "по-честному" добавить в трассу реальное время непросто

Но как правило, система изменяет своё состояние **не** континуально часто, а в худшем случае счётно

Чтобы анализировать СРВ, достаточно обозревать те моменты времени, в которые система изменяет своё состояние

Системы реального времени

А почему бы не абстрагироваться от времени и не рассматривать только такие системы и свойства, как были раньше в лекциях?

В СРВ, как правило, существуют директивные сроки выполнения задач, и для таких систем требуются гарантии выполнения этих сроков:

- птица не поела вовремя — умерла
- парашют не раскрылся в срок — тоже не очень хорошо
- подушка безопасности раскрылась на долю секунды позже — и опять всё плохо

Упомянутые здесь системы — это жёсткие СРВ: для таких систем требуется обязательное соблюдение директивных сроков

Системы реального времени

Есть двойственное понятие: мягкие СРВ

Срыв директивных сроков в таких системах возможен, но как правило ведёт к ухудшению качества работы системы

Например, обычная операционная система на высоком уровне абстракции — это мягкая СРВ: если процесс освободил память на секунду позже запланированного, то система продолжит работать, но некоторое время будет работать медленнее, чем могла бы

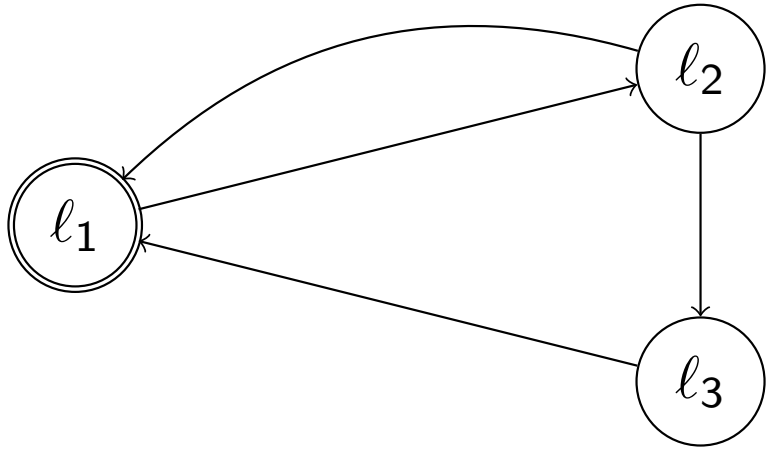
Для анализа жёстких и мягких СРВ используются совершенно разные подходы и модели

Мы будем рассматривать только **жёсткие** СРВ

Временные автоматы

— это самый популярный формализм, используемый при верификации жёстких СРВ

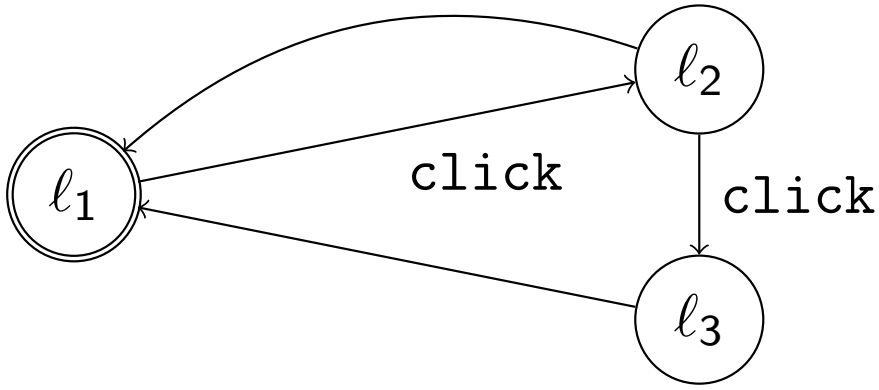
Прежде чем вводить формальные определения, рассмотрим такой **пример:**



Это **автомат**, с помощью которого мы попытаемся промоделировать щелчок и двойной щелчок мышки:

- ℓ_1: никаких щелчков не поступало
- ℓ_2: поступил щелчок
- ℓ_3: после щелчка поступил второй щелчок

Временные автоматы

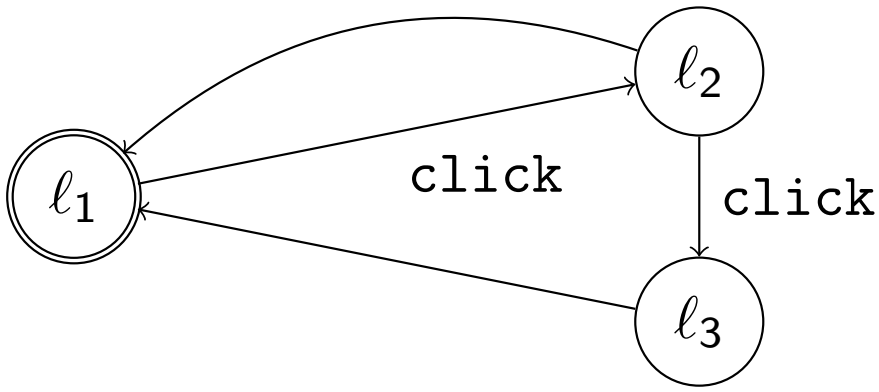


Что нужно добавить в автомат, чтобы он адекватно описывал двойной щелчок мышкой?

Автомат взаимодействует с **окружением**: нужно уметь различать, когда пришёл щелчок мыши, и адекватно на него реагировать

Так в автомате на переходах появляются **события**; здесь — событие `click`

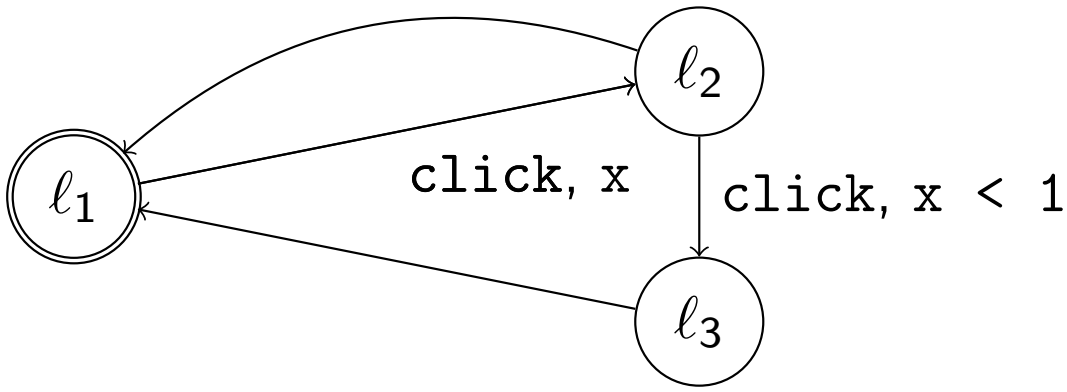
Временные автоматы



Чтобы два нажатия на кнопку мышки трактовались как двойной щелчок, между первым и вторым нажатиями должно пройти *достаточно мало* времени — скажем, меньше секунды

Чтобы уметь отслеживать, сколько времени прошло после первого щелчка, заведём **таймер** `x`: переменную, значение которой принимает действительные значения и увеличивается с той же скоростью, с которой течёт время

Временные автоматы

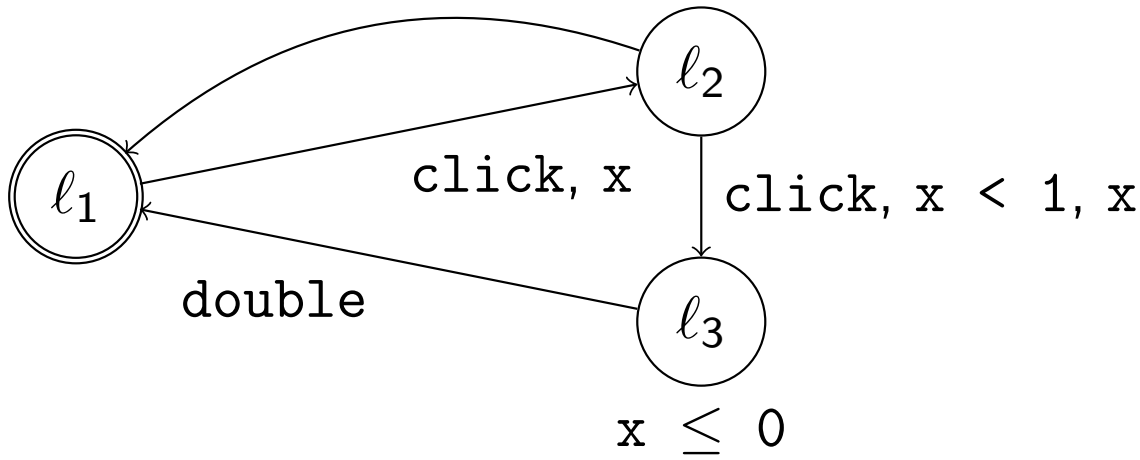


Как только в системе возникло событие `click`, эта переменная должна быть **сброшена** в 0: начинаем отслеживать, сколько времени прошло с первого щелчка

Если второе событие `click` произошло меньше чем через секунду после первого, то следует перейти в состояние двойного щелчка

Это ограничение на возможность срабатывания самого правого перехода: **предусловие** `x < 1`

Временные автоматы

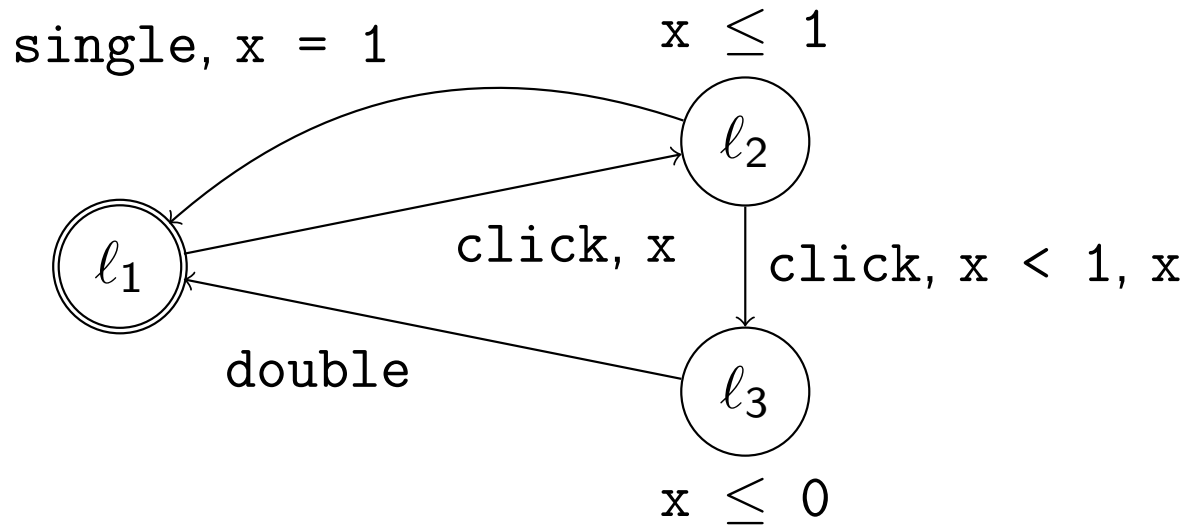


Из состояния “был двойной щелчок” следует *немедленно* перейти в состояние “ожидаем первый щелчок” и проинформировать окружение о том, что произошёл двойной щелчок

Для этого

- сбросим $\mathtt{x}$ при переходе в ℓ_3
- запретим в ℓ_3 течь времени: пометим **инвариантом** $\mathtt{x} \leq 0$ (*при нахождении в этом состоянии значение переменной x не может быть больше* 0)
- при переходе в ℓ_1 произведём событие `double`

Временные автоматы



Если мы прождали в состоянии одинарного щелчка секунду и событие `click` не произошло, то следует *немедленно* сделать заключение, что произошёл одинарный щелчок мышью

Для этого добавим инвариант $x \leq 1$ в состояние одинарного щелчка, разрешим перейти из этого состояния в исходное только в том случае, если $x = 1$, и при переходе в исходное состояние произведём событие `single`

В результате получен **временной автомат**, распознающий одинарный и двойной щелчки мыши

Временные автоматы

А какой спектр действительных констант можно использовать в описании автомата?

Если цель — описать **алгоритм** верификации временных автоматов, то использование **континуума** различных констант помешает достигнуть этой цели: входные данные алгоритма должны быть **конечными**, а конечного способа описания произвольных действительных чисел не существует

Временные автоматы

Чтобы ограничиться счётным набором констант, не снизив выразительные возможности модели, обычно предлагаются такие рассуждения:

- время наступления событий в СРВ всегда измеряется с погрешностью
- значит, вместо действительного числа во временных ограничениях можно использовать *достаточно близкое* рациональное число
- когда модель составлена и образовалось множество используемых в модели рациональных чисел, можно привести эти числа к общему знаменателю и "масштабировать" время модели, убрав знаменатель
- в конечном итоге можно без снижения выразительных возможностей использовать в модели **только целые неотрицательные числа**

Временные автоматы

Временной автомат — это система $(L, \ell_0, \Sigma, C, I, T)$, где

- L — конечное множество состояний автомата
- $\ell_0 \in L$ — начальное состояние
- Σ — конечное множество событий
- C — конечное множество таймеров
- $I : L \to inv(C)$ — разметка состояниий инвариантами
- $T \subseteq L \times (\Sigma \cup \{\lambda\}) \times guard(C) \times 2^C \times L$ — отношение переходов
  - λ — особое событие, означающее отсутствие событий
  - третья компонента перехода — предусловие
  - четвёртая компонента перехода — множество сбрасываемых таймеров

Временные автоматы

Если разрешить в качестве предусловий и инвариантов писать очень выразительные формулы, то задача анализа поведения временного автомата окажется неоправданно сложной

Чтобы упростить эту задачу, оставив достаточно широкий спектр описательных возможностей, в модели временных автоматов предлагаются такие множества $inv(C)$, $guard(C)$:

- элементарные ограничения $ATC(C)$ — это выражения **true**, **false** и всевозможные выражения вида $x \bowtie k$ и $x - y \bowtie k$, где
  - x, y — таймеры
  - k — целая неотрицательная константа (то есть $k \in \mathbb{N}_0$)
  - $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$
- в множество $guard(C)$ входят всевозможные формулы, составленные из элементарных ограничений и логических связок &, $\vee$, $\neg$, $\rightarrow$
- в множество $inv(C)$ входят всевозможные конъюнкции элементаных ограничений **true**, $c < k$, $c \leq k$

Семантика временных автоматов

Содержательное описание

Текущее состояние вычисления (*конфигурация*) временного автомата определяется тем, в каком состоянии находится автомат и какие значения имеют его таймеры.

Изменение состояния компонента СРВ занимает некоторое время, и в течение этого времени компонент находится в “промежуточном” состоянии: одно состояние уже покинуто, а другое ещё не достигнуто

Семантика временных автоматов

Содержательное описание

В семантике временного автомата такое поведение компонентов системы невозможно, вместо это автомату разрешается выполнение одного из двух действий:

- некоторое время находиться в некотором состоянии, не изменяя его, если это позволяет инвариант состояния
- мгновенно изменить состояние согласно переходу с истинным предусловием, сбросив таймеры, указанные на переходе и перейдя в состояние с истинным инвариантом

При моделировании СРВ следует учитывать такое расхождение “реальной” и “модельной” семантик

Семантика временных автоматов

Более строгое описание

Конфигурация временного автомата $(L, \ell_0, \Sigma, C, I, T)$ — это пара (ℓ, d), где $\ell \in L$ и $d : C \to \mathbb{R}_{\geq 0}$

($\mathbb{R}_{\geq 0}$ — *множество всех неотрицательных действительных чисел*)

Для простоты иногда будем полагать, что таймеры автомата упорядочены: $C = (x_1, \ldots, x_m)$ — и записывать конфигурацию автомата как состояние, после которого располагаются значения всех таймеров: $(\ell, k_1, \ldots, k_m)$

Начальная конфигурация автомата имеет вид $(\ell_0, 0, 0, \ldots, 0)$

Семантика временных автоматов

Более строгое описание

Шаг вычисления временного автомата — это изменение конфигурации (ℓ, d) на конфигурацию (ℓ', d') одним из двух способов:

- продвижение времени: $(\ell, d) \rightarrow (\ell', d')$
  - $\ell' = \ell$
  - существует константа $k \in \mathbb{R}_{>0}$, такая что $\forall x (x \in C \rightarrow d'(x) = d(x) + k)$
    - более короткая запись: $d' = d + k$
    - $\mathbb{R}_{>0}$ — множество всех положительных действительных чисел
  - инвариант $I(\ell)$ выполнен для значений таймеров d'
    - более короткая запись: $d' \models I(\ell)$
- изменение состояния: $(\ell, d) \rightarrow (\ell', d')$
  - в автомате есть переход $(\ell, \sigma, g, C, \ell')$
  - $d \models g$
  - $d'(x) = 0$ для $x \in C$ и $d'(y) = d(y)$ для $y \notin C$
    - более короткая запись: $d' = reset(d, C)$
  - $d' \models I(\ell')$

Семантика временных автоматов

Более строгое описание

Вычисление временного автомата — это бесконечная последовательность конфигураций, в которой следующая конфигурация получается из предыдущей продвижением времени или изменением состояния

Неправдоподобные вычисления

Если разрешить временному автомату продвигать время и изменять состояния произвольным образом, то немедленно возникают вычисления автомата, *неадекватные* по отношению к поведению реальных СРВ

Стагнация системы (*livelock*)

Возможна ситуация, в которой автомат всегда может изменить состояние, но вместо этого постоянно продвигает время:

$$(\ell, 0) \to (\ell, 1) \to (\ell, 2) \to (\ell, 3) \to \dots$$

В реальной системе действуют "законы справедливости": ситуация, в которой состояние может быть изменено, но никогда не изменяется, считается неправдоподобной

Неправдоподобные вычисления

Если разрешить временному автомату продвигать время и изменять состояния произвольным образом, то немедленно возникают вычисления автомата, *неадекватные* по отношению к поведению реальных СРВ

Конвергентные вычисления

Для временного автомата возможна следующая ситуация: он сможет изменить состояние, если пройдёт некоторое время (скажем, единица времени), однако в бесконечном вычислении единица времени никогда не проходит:

$$(\ell, 0) \to (\ell, \tfrac{1}{2}) \to (\ell, \tfrac{3}{4}) \to (\ell, \tfrac{7}{8}) \to \dots$$

Для реальной системы такая ситуация невозможна: любой момент времени должен рано или поздно наступить

Неправдоподобные вычисления

Если разрешить временному автомату продвигать время и изменять состояния произвольным образом, то немедленно возникают вычисления автомата, *неадекватные* по отношению к поведению реальных СРВ

Вычисления Зенона

Возможна и более "хитрая" вариация конвергентности вычисления: автомат бесконечное число раз как изменяет состояние, так и продвигает время, однако существует граница протёкшего времени, которая никогда не достигается:

$$(\ell, 0) \to (\ell, \tfrac{1}{2}) \to (\ell', 0) \to (\ell', \tfrac{1}{4}) \to (\ell, 0) \to \dots$$

Такие вычисления служат иллюстрацией *апорий Зенона*, суть которых сводится к выполнению *бесконечного* числа действий в *конечное* время

Такие вычисления также считаются неправдоподобными (*как и апории Зенона*)

Неправдоподобные вычисления

И как же исключить неправдоподобные вычисления из семантики временного автомата?

Почти все неправдоподобные вычисления исключаются следующим ограничением:

- автомату запрещено два раза подряд продвигать время

Стагнация системы:

$$(\ell, 0) \to (\ell, 1) \to (\ell, 2) \to \ldots$$ — невозможна

Конвергентное вычисление:

$$(\ell, 0) \to (\ell, \tfrac{1}{2}) \to (\ell, \tfrac{3}{4}) \to \ldots$$ — невозможно

Вычисление Зенона:

$$(\ell, 0) \to (\ell, \tfrac{1}{2}) \to (\ell', 0) \to (\ell', \tfrac{1}{4}) \to \ldots$$ — возможно

Модель в формализме временных автоматов, содержащая вычисления Зенона, — это некорректно построенная модель

Семантика временных автоматов

Ранее в лекциях семантика каждой системы в конечном итоге описывалась моделью Крипке

А можно ли сделать так и для временных автоматов?

Модель Крипке $M(A)$ для временного автомата A определяется так:

- состояния модели Крипке — это всевозможные конфигурации автомата
- переходами связаны пары конфигураций, отличающихся одним шагом вычисления
- начальное состояние модели Крипке — это начальная конфигурация автомата

Семантика временных автоматов

А какими высказываниями размечены состояния этой модели Крипке?

Состояние модели Крипке помечается всевозможными выполненными в этом состоянии *элементарными ограничениями*, к которым при разметке добавляются атомы вида $A.\ell$: "автомат A находится в состоянии ℓ"
(*множество всех атомов вида* $A.\ell$ *обозначим записью* $ALC(A)$)

А будет ли отношение переходов модели Крипке тотальным?

Не обязательно, но временной автомат, модель Крипке которого нетотальна, — это **некорректно построенный автомат**

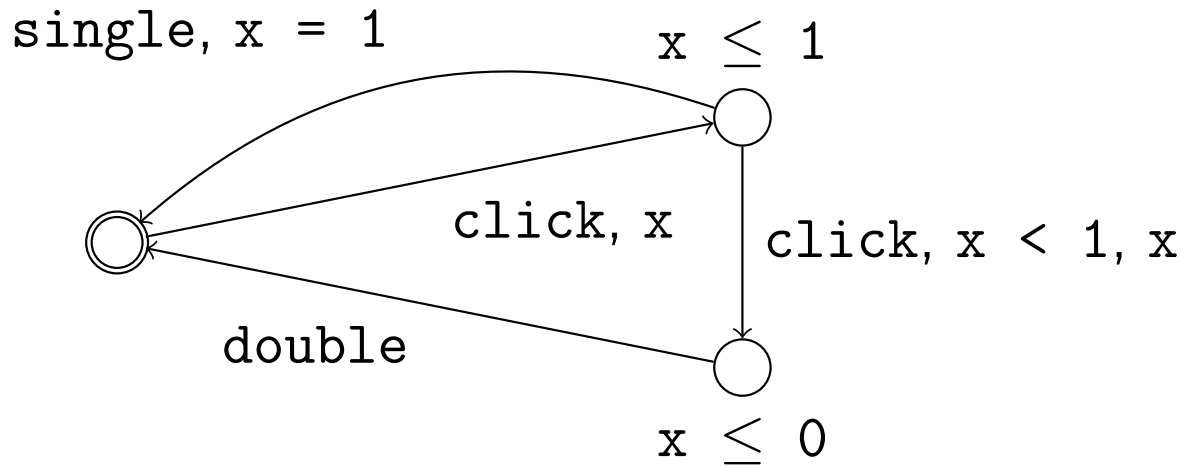
А как быть с чередованием типов шагов вычисления?

Можно расценивать это как особый вид справедливости, специфичный для временных автоматов

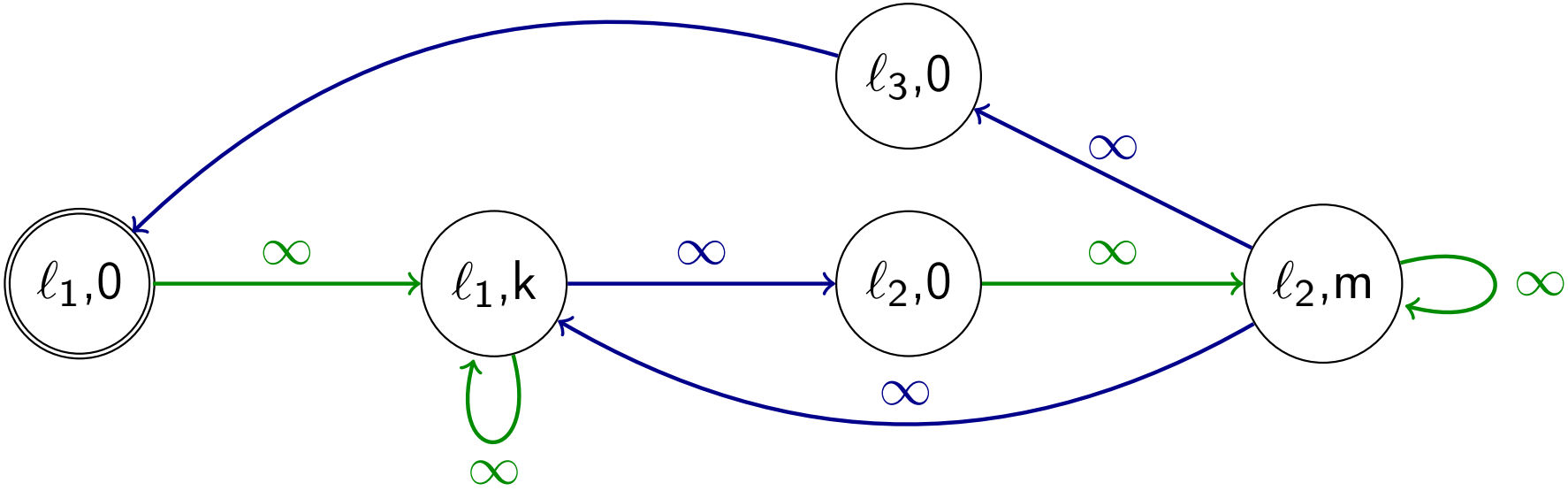
А действительно ли это модель Крипке?

Не совсем: модель Крипке для временного автомата в общем случае **бесконечна**

Семантика временных автоматов



Модель Крипке для этого автомата будет выглядеть так:



Композиция временных автоматов

СРВ, описанная в формализме временных автоматов, как правило состоит из *нескольких* временных автоматов, взаимодействующих между собой посредством событий

Например, параллельно автомату, распознающему щелчок и двойной щелчок мышки, в системе может присутствовать автомат, моделирующий поведение пользователя, щёлкающего мышкой

Автомат пользователя и автомат, распознающий щелчки, связываются между собой через событие щёлчка мышкой

Композиция временных автоматов

Как правило, при взаимодействии автоматов через события один автомат *генерирует* событие, тогда как другой автомат *принимает* это событие и *обрабатывает* его

Например, пользователь *генерирует* событие щелчка мышкой, тогда как компьютер *принимает* и *обрабатывает* это событие

Такое использование событий может быть переформулировано следующим образом:

- между автоматами существует канал взаимодействия *ch*
- генерация — это событие посылки сигнала в канал *ch*: `ch!`
- приём и обработка — это событие приёма сигнала из канала *ch*: `ch?`

Композиция временных автоматов

Самый распространённый тип взаимодействия автоматов через каналы, — это синхронное взаимодействие типа точка-точка (рандеву)

Рандеву временных автоматов происходит следующим образом:

- выбирается автомат, согласно предусловию перехода способный послать сигнал в канал взаимодействия
- выбирается автомат, согласно предусловию перехода способный принять сигнал из канала
- если при сбросе всех таймеров, размечающих два выбранных перехода, инварианты новых состояний автоматов выполнены, то выбранные переходы одновременно выполняются
- в противном случае выбранная пара переходов не может быть выполнена

Сети временных автоматов

Сеть временных автоматов — это система $(C, Chan, (A_1, \ldots, A_k))$, где

- C — конечное множество общих таймеров
- $Chan$ — конечное множество общих каналов взаимодействия
- $A_i = (L^i, \ell_0^i, \{ch!, ch? \mid ch \in Chan\}, C, I^i, T^i)$ — временной автомат над общими таймерами и общими каналами взаимодействия

Автоматы сети работают *асинхронно* согласно *семантике чередующихся вычислений*

Единственная возможность *синхронной* работы автоматов — это рандеву

Сети временных автоматов

Конфигурация сети $(C, Chan, (A_1, \ldots, A_m))$,
$A_i = (L^i, \ell_0^i, \Sigma, C, I^i, T^i)$, — это система $(\ell_1, \ldots, \ell_m, d)$, где
$\ell_i \in L^i$ и $d : C \to \mathbb{R}_{\geq 0}$

Начальная конфигурация сети имеет вид $(\ell_0^1, \ldots, \ell_0^m, 0, 0, \ldots, 0)$

Шаг вычисления сети — это изменение конфигурации $(\ell_1, \ldots, \ell_k, d)$ на конфигурацию $(\ell'_1, \ldots, \ell'_m, d')$ одним из трёх способов:

1. продвижение времени:
   - $\ell'_1 = \ell_1, \ldots, \ell'_k = \ell_m$
   - существует константа $k \in \mathbb{R}_{>0}$, такая что $d' = d + k$
   - $d' \models I^1(l_1) \,\&\, \ldots \,\&\, I^m(l_m)$

Сети временных автоматов

Конфигурация сети $(C, Chan, (A_1, \ldots, A_m))$,
$A_i = (L^i, \ell_0^i, \Sigma, C, I^i, T^i)$, — это система $(\ell_1, \ldots, \ell_m, d)$, где
$\ell_i \in L^i$ и $d : C \to \mathbb{R}_{\geq 0}$

Начальная конфигурация сети имеет вид $(\ell_0^1, \ldots, \ell_0^m, 0, 0, \ldots, 0)$

Шаг вычисления сети — это изменение конфигурации $(\ell_1, \ldots, \ell_k, d)$ на конфигурацию $(\ell'_1, \ldots, \ell'_m, d')$ одним из трёх способов:

2. асинхронное изменение состояния автомата A_i:
   - $\ell'_p = \ell_p$ для $p \neq i$
   - $(\ell_i, \lambda, g, \mathcal{C}, \ell'_i) \in T^i$
   - $d \models g$
   - $d' = reset(d, \mathcal{C})$
   - $d' \models I^i(\ell'_i)$

Сети временных автоматов

Конфигурация сети $(C, Chan, (A_1, \ldots, A_m))$,
$A_i = (L^i, \ell_0^i, \Sigma, C, I^i, T^i)$, — это система $(\ell_1, \ldots, \ell_m, d)$, где
$\ell_i \in L^i$ и $d : C \to \mathbb{R}_{\geq 0}$

Начальная конфигурация сети имеет вид $(\ell_0^1, \ldots, \ell_0^m, 0, 0, \ldots, 0)$

Шаг вычисления сети — это изменение конфигурации $(\ell_1, \ldots, \ell_k, d)$ на конфигурацию $(\ell'_1, \ldots, \ell'_m, d')$ одним из трёх способов:

3. синхронное изменение состояний автоматов A_i, A_j:
   - $\ell'_p = \ell_p$ для $p \neq i$, $p \neq j$
   - $(\ell_i, c!, g', C', \ell'_i) \in T^i$
   - $(\ell_j, c?, g'', C'', \ell'_j) \in T^j$
   - $d \models g' \,\&\, g''$
   - $d' = reset(d, C' \cup C'')$
   - $d' \models I^i(\ell'_i) \,\&\, I^j(\ell'_j)$

Сети временных автоматов

Определения вычисления сети и модели Крипке $A(N)$ для сети N дословно совпадают с теми же определениями для одного автомата

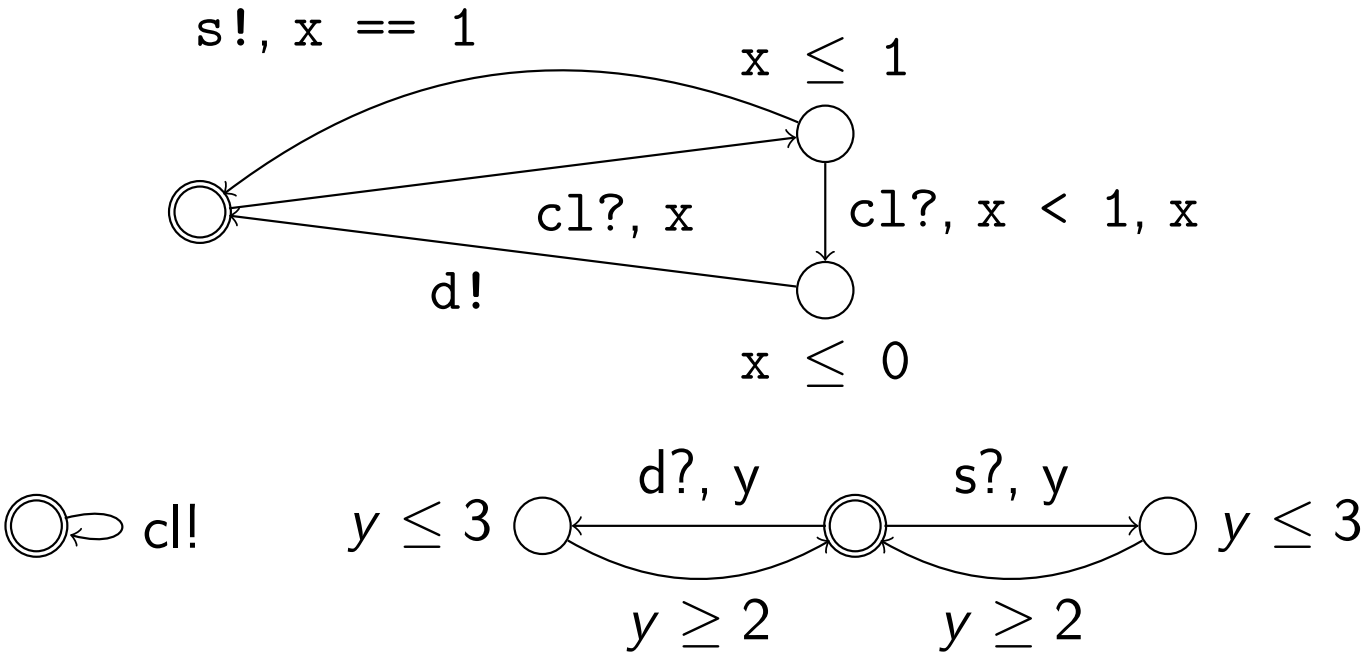
Как и для временного автомата, для сети остаётся неустранимая проблема наличия *вычислений Зенона* и других неправдоподобных вычислений

Пример

Попробуем добавить к автомату, распознающему одинарный и двойной щелчки мыши,

- автомат, моделирующий поведение пользователя: произвольным образом генерирующий события щелчка мыши
- автомат, обрабатывающий одинарный и двойной щелчки: на обработку каждого щелчка тратится от 2 до 3 единиц времени

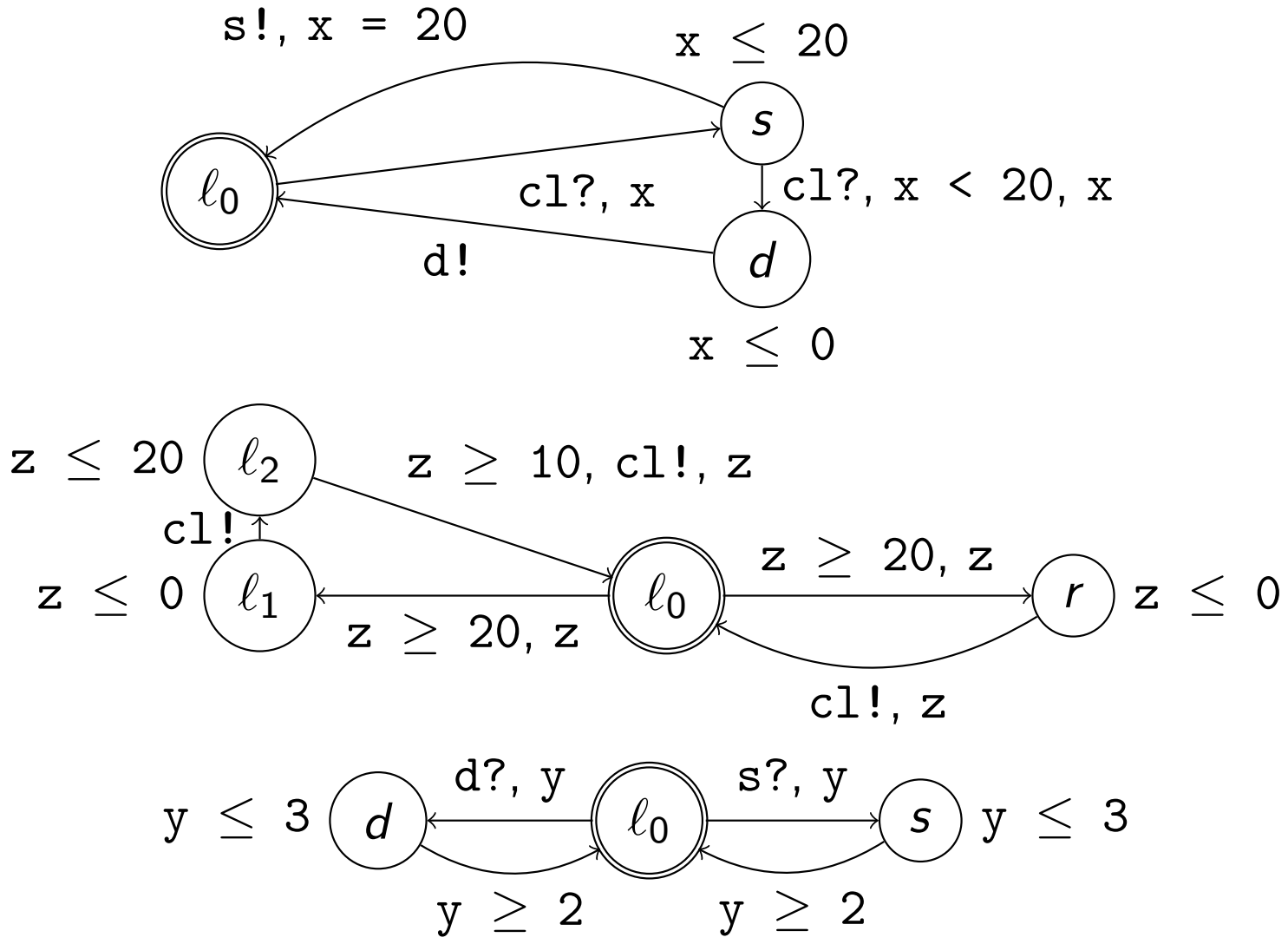
Сети временных автоматов



Корректно ли работает эта система?

Нет, а почему?

Сети временных автоматов



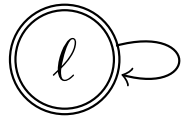
А если сделать пользователя более “разумным” и медленным, то система будет работать корректно

Timed CTL

Формальная семантика автоматов и сетей — это модель Крипке, а значит, можно попытаться сформулировать требования к этим системам в виде CTL-формул

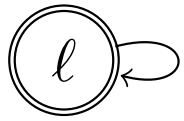
При попытке это сделать “в лоб” возникает проблема несоответствия формального смысла формул тому, что содержательно вкладывается в эти формулы

Эту проблему можно проиллюстрировать таким **примером**:



Предъявим к этой системе такое требование: $\mathbf{AF}x = 1$

Timed CTL



$\varphi : \mathbf{AF}x = 1$

Тот, кто формулировал это требование, *скорее всего* вкладывал в него такой содержательный смысл:

как бы ни работала система,
рано или поздно наступит момент времени 1

Значит, это требование *должно быть* верным

При этом существует, например, такое вычисление автомата:

$$(\ell, 0) \to (\ell, 2) \to \dots$$

В этом вычислении момент времени 1 никогда не наступает явно, а значит, требование φ не выполнено для автомата

Чтобы устранить такое несоответствие содержательного и формального смыслов формул, следует адаптировать семантику CTL к работе в реальном времени

Timed CTL

Синтаксис формул логики TCTL (Timed CTL) для автомата $A = (L, \ell_0, \Sigma, C, I, T)$ [сети $N = (C, Chan, (A_1, \ldots, A_m))$] совпадает с синтаксисом формул логики CTL **без оператора X** над множеством атомарных высказываний $ATC(C) \cup ALC(A)$ $[ATC(C) \cup ALC(A_1) \cup \cdots \cup ALC(A_m)]$

При этом **семантики** TCTL- и CTL-формул различаются

Перед тем как описать различия этих семантик, введем одно техническое понятие

Для шага вычисления $(\ell, d) \to (\ell', d')$ допустимым продвижением времени назовём константу $k \in \mathbb{R}_{\geq 0}$, такую что:

- если $d' = d + p$, то $0 \leq k \leq p$
- если d' получается из d сбросом некоторых таймеров, то $k = 0$

Timed CTL

Рассмотрим вычисление временного автомата:

$$tr = (\ell_0, d_0) \to \cdots \to (\ell_i, d_i) \to \ldots$$

Тогда $tr \models \varphi \mathbf{U} \psi$ (*в семантике TCTL*) в том и только том случае, если существуют конфигурация (ℓ_i, d_i) и допустимое продвижение k для этой конфигурации и следующей, такие что

- $(\ell_i, d_i + k) \models \psi$
- для любой конфигурации (ℓ_j, d_j), $j < i$, и для любого допустимого продвижения времени k' для j-й и $(j+1)$-й конфигураций верно $(\ell_j, d_j + k') \models \varphi$
- для любого k', $0 \leq k' < k$, верно $(\ell_i, d_i + k') \models \varphi$

Семантика остальных темпоральных операторов задаётся естественным образом: $\mathbf{F}\varphi = \mathbf{true}\mathbf{U}\varphi$, $\mathbf{G}\varphi = \neg\mathbf{F}\neg\varphi$

Выполнимость формулы φ в модели M в изменённой таким образом семантике будем обозначать так: $M \models_{TCTL} \varphi$

Timed CTL

Примеры требований, предъявляемых к сети, состоящей из распознавателя щелчков мыши R, пользователя U и обработчика щелчков H:

- $\mathbf{AG}(!H.\ell_0 \to \mathbf{AF}H.\ell_0)$: обработчик щелчков обязательно завершает обработку каждого набора щелчков
- $\mathbf{AG}(!R.\ell_0 \to x \leq 20)$: таймер распознавателя щелчков не поднимается выше 20 единиц времени при обработке щелчка
- $\mathbf{EF}H.d$: пользователь может добиться того, чтобы система распознала его работу с мышью как двойной щелчок
- $\mathbf{AG}(z - x \geq 0)$: таймер z не может обнулиться, если $x \neq 0$

Конец лекции 9