Задания выбираются путём запроса желаемого номера задания (в диапазоне 1-36) на мою почту garanina@iis.nsk.su . Можно присылать несколько номеров на случай, если какие-то будут уже заняты. Обновление свободных номеров по мере возможности.

Задания состоят из двух частей -- теоретическая (задачи 1 и 2) и практическая (задача 3). Сложность задач разная, поэтому непростые практические задачи уравновешиваются лёгкими теоретическими.

Для получения оценки должны быть выполнены все задачи. Теоретическая и практическая части оцениваются отдельно по 5-ти балльной системе.

Выполненные задания присылать мне на почту до 15.04. Теоретическая часть – файл Word и подобные, можно с картинками. Практическая часть – файл Promela с моделью и свойствами программы и описание полученных результатов в свободной форме. В случае затруднения с оценкой может понадобиться личная встреча. Возможны консультации.

Задача 1.

Записать высказывание естественного языка формулой темпоральной логики. Обосновать запись.

Задача 2. http://patterns.projects.cs.ksu.edu/documentation/patterns.shtml

Заданы шаблон требования, его область действия и логика его семантики.
Для этих данных придумать:

1. конкретное свойство системы;
2. выразить его в соответствующей логике;
3. построить для этого свойства систему (модель Крипке):
    - система должна содержать не менее 4 состояний и отношение переходов должно быть недетерминированным;
4. исследовать выполнимость свойства в построенной системе, применяя алгоритм проверки моделей.

Задача 3.

Практическая верификация в системе SPIN http://spinroot.com/spin/whatispin.html

1. Написать на языке PROMELA модель системы.
2. Определить 2-3 свойства.
3. Записать свойства в виде формул LTL/assert.
4. Выполнить верификацию в системе iSpin.
5. Модифицировать модель/свойство таким образом, чтобы одно из проверяемых свойств не выполнялось.
6. Проанализировать контр-пример и объяснить, почему не выполняется свойство.

ЗАДАНИЕ 1

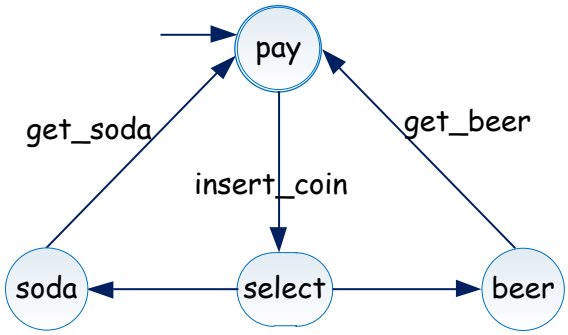
1. **Владислав Швец**

   И вот я жду, готовлюсь к чуду,
   Сперва – с трудом, затем – шутя.
   И если выйдет… нет, не буду.
   А может быть, ещё… хотя…

2. Универсальность; Между; CTL.

3. Автомат для выдачи напитков

Система переходов на рисунке моделирует автомат для выдачи напитков. Пользователь вставляет монетиу, после этого автомат недетерминированно выбирает выдаёт пиво или содовую и затем выдаёт выбранный напиток.



ЗАДАНИЕ 2

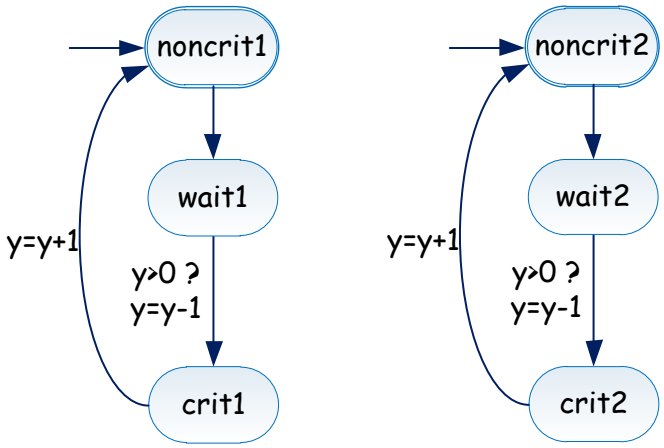
1. **«Москва — Петушки» — поэма в прозе Венедикта Ерофеева.**

   Я бы согласился жить на земле вечно, если прежде мне показали бы уголок, где не всегда есть место подвигу…

2. Отсутствие; После-пока; CTL.

3. Взаимное исключение с помощью семафоров

Процессы P_1 и P_2 представлены системами переходов на рисунке. Они совместно используют двоичный семафор y. Когда y = 0 это указывает на то, что семафор - блокировка для доступа к критической секции - в настоящее время занят одним из процессов. Когда y = 1, семафор свободен. Система переходов процессов представлены на рисунке. Для простоты локальные переменные и общие переменные, отличные от y, не рассматриваются. Кроме того, действия внутри и снаружи критических секций опущены. Состояния $noncrit_i$ представляют некритические действия, $wait_i$ моделируют ситуацию, в которой P_i ждет входа в критическую секцию, и $crit_i$ моделируют пребывание в критической секции.



ЗАДАНИЕ 3

1. **Книга Екклезиаста**

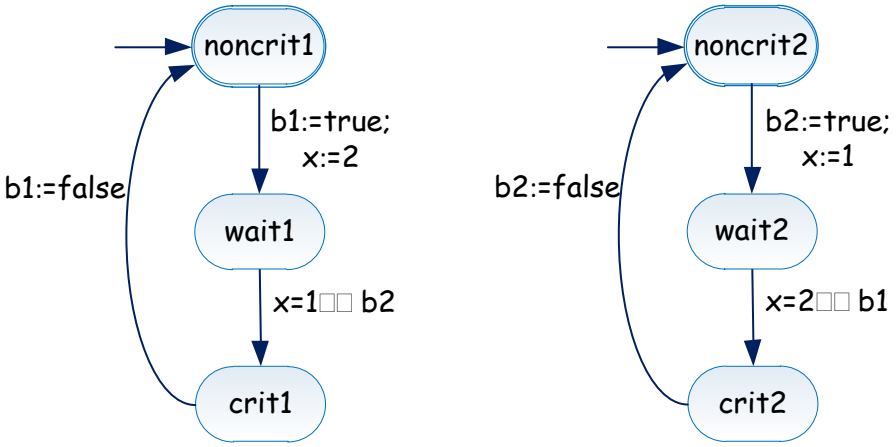
   …всё произошло из праха и всё возвратится в прах.

2. Существование; После-пока; CTL.

3. Взаимное исключение Питерсона

Рассмотрим процессы P_1 и P_2 с общими переменными b_1, b_2 и x. b_1 и b_2 являются булевыми переменными, а x∈{1, 2}. Стратегия планирования реализуется с использованием x следующим образом. Если оба процесса хотят войти в критическую секцию (т. е. находятся в состоянии $wait_i$), значение переменной x определяет, какой из двух процессов может войти в его критическую секцию: если $x = i$, то P_i может войти в свою критическую секцию (для i = 1, 2). При входе в позицию $wait_1$ процесс P_1 выполняет x: = 2, тем самым давая право на обработку P_2 для входа в критическую секцию. Таким образом, значение x указывает, чья очередь входа в критическую секцию. Симметрично, P_2 устанавливает x в 1, когда начинает ждать. Переменные b_i предоставляют информацию о текущем местоположении P_i. Точнее, $b_i = wait_i$ ∨ criti. b_i устанавливается, когда P_i начинает ждать. В псевдокоде выполнения P_1 выглядит следующим образом (для процесса P_2 аналогично):

```
P1 loop forever
… (* noncritical actions *)
b1 := true; x := 2; (* request *)
wait until (x = 1 ∨ ¬b2)
do critical section od
b1 := false (* release *)
... (* noncritical actions *)
end loop
```



ЗАДАНИЕ 4

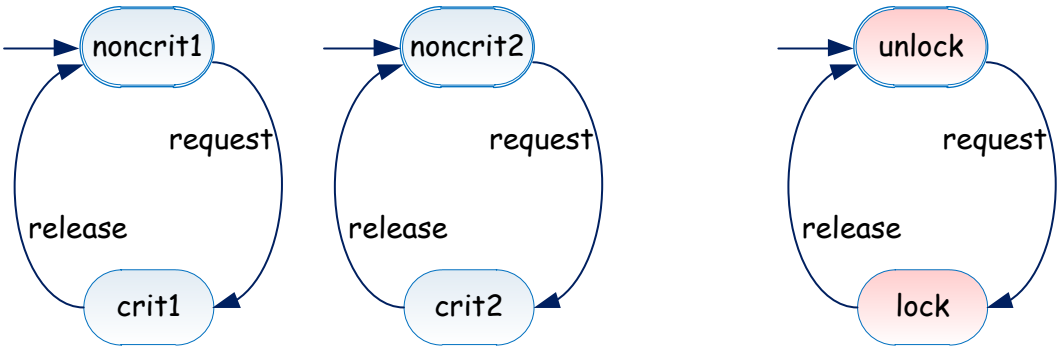
1. **Евангелие от Луки**

   Сын Человеческий, придя, найдёт ли веру на земле?

2. Предшествование; Перед; LTL.

3. Взаимное исключение с арбитром

Взаимное исключение между процессами P_1 и P_2 заключается в моделировании двоичного семафора, который регулирует доступ к критической секции посредством отдельного параллельного процесса *Arbiter*, который взаимодействует с P_1 и P_2 посредством хэндшейкинга для синхронизации действий. Для простоты мы игнорируем фазу ожидания и предполагаем, что P_i чередуется бесконечно часто между некритическими и критическими секциями. Упрощённые системы переходов процессов имеют только два состояния: $crit_i$ и $noncrit_i$. Процесс *Arbiter* имитирует двоичный семафор. P_1 и P_2 общаются с *Arbiter* для синхронизации действий *request* и *release*.



ЗАДАНИЕ 5

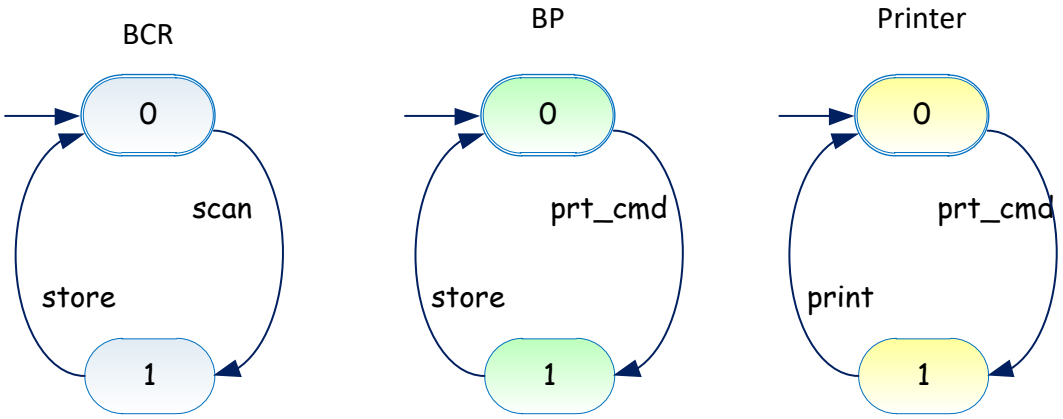
1. **Хагакурэ**

   Если каждое утро и каждый вечер ты будешь готовить себя к смерти и сможешь жить так, словно твое тело уже умерло, ты станешь подлинным самураем.

2. Ответ; Перед; CTL.

3. Система бронирования.

Систему бронирования на кассе супермаркета состоит из трех процессов: считывателя штрих-кода *BCR*, фактической программы бронирования *BP* и принтера *Printer*. Считыватель штрих-кода *BCR* считывает штрих-код и передает данные только что отсканированного продукта в программу бронирования *BP*. При получении данных программа бронирования *BP* передает цену товара на принтер *Printer*, который печатает идентификатор товара и цену на чек. Синхронное взаимодействие между считывателем штрих-кода *BCR* и программой бронирования *BP*, а также между программой бронирования *BP* и принтером *Printer* выполняется посредством хэндшейкинга. Каждый процесс находится в одном из двух состояний 0 и 1.



ЗАДАНИЕ 6

1. **Хагакурэ**

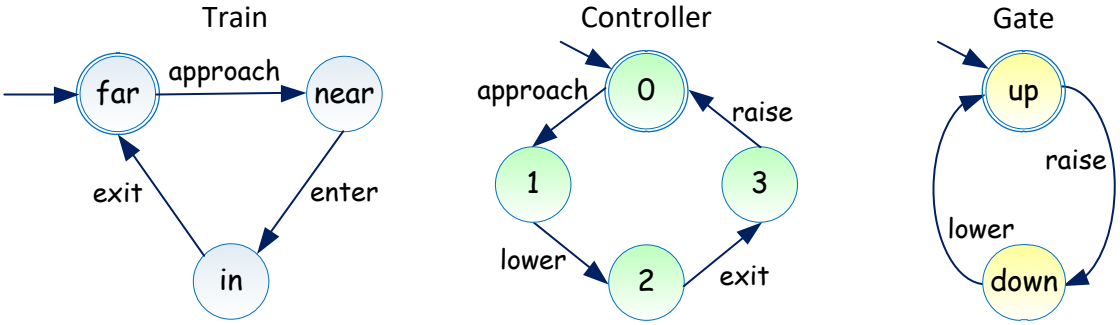
   Если боги не услышат мои молитвы только потому, что я осквернён кровью, я убеждён, что ничего не могу поделать с этим и поэтому продолжаю молиться, невзирая на осквернённость.

2. Существование; Между; CTL.

3. Железнодорожный переезд

Для железнодорожного переезда необходимо разработать систему управления, которая по получении сигнала, указывающего, что поезд приближается, закрывает шлагбаум и открывает его только после того, как поезд отправил сигнал, указывающий, что он пересек дорогу. Система состоит из трех компонентов: *Train*, *Gate* и *Controller*.

Для простоты предполагаем, что все поезда проходят соответствующий участок дороги в том же направлении - слева направо. Состояния системы переходов для *Train* имеют следующий интуитивный смысл: в состоянии *far* поезд далеко от переезда, в состоянии *near* он приближается к переходу и только что отправил сигнал, чтобы уведомить об этом, и в состоянии *in* находится на переезде. Интерпретация состояний *Gate* очевидна. *Controller* синхронизирует изменения своих состояний с *Train* (относительно действий *approach* и *exit*) и с *Gate* (относительно действий *lower* и *raise*).



ЗАДАНИЕ 7

1. **Андрей Платонов**

   Шло большое звёздное время, что безвозвратно проходит, считая свои отмирающие части.

2. Универсальность; Перед; LTL.

3. Протокол альтернирующего бита (ABP)
Простой протокол передачи данных, который повторно передает потерянные или поврежденные сообщения.

Сообщения отправляются от передатчика A к приемнику B. Предположим, что канал от A до B инициализирован и что нет сообщений в пути. Каждое сообщение содержит часть данных, контрольную сумму и однобитовый порядковый номер, то есть значение, которое равно 0 или 1.

Когда A отправляет сообщение, он отправляет его непрерывно с тем же порядковым номером, пока он не получит подтверждение подтверждения *ACK* из B, которое содержит тот же порядковый номер. Когда это произойдет, A дополняет (переворачивает) порядковый номер и начинает передачу следующего сообщения.

Когда B получает сообщение от A, он проверяет контрольную сумму. Если сообщение не повреждено, B отправляет обратно *ACK* с тем же порядковым номером. Если это первое сообщение с этим порядковым номером, оно отправляется для обработки. Последующие сообщения с одним и тем же битом последовательности просто подтверждаются. Если сообщение повреждено, B отправляет обратно отрицательное подтверждение *NAK*. Это необязательно, так как A продолжит передачу, пока не получит правильный *ACK*.

A обрабатывает поврежденные сообщения *ACK* и сообщения *NAK* таким же образом. Самое простое поведение - игнорировать их все и продолжить передачу.

ЗАДАНИЕ 8

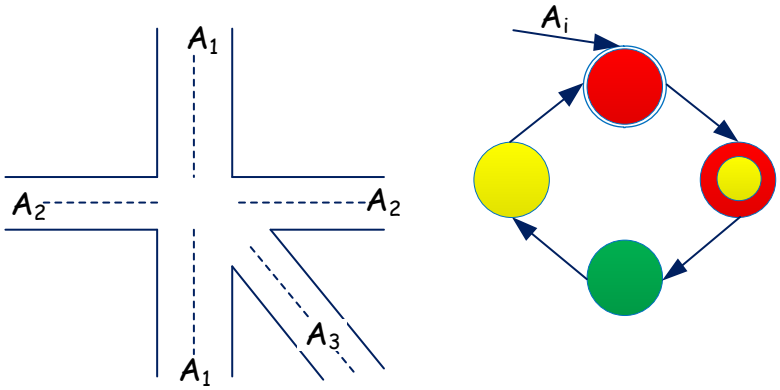
1. **Планета людей**

Ждал, что жизнь подаст мне знак,- и не дождался.

2. Существование; Перед; LTL.

3. Светофор.

Построить систему светофоров, которые включают зеленый свет в следующем порядке: A1, A2, A3, A1, A2, A3, ….



ЗАДАНИЕ 9

1. **Маленький принц**

   Заслышав людские шаги, я всегда убегаю и прячусь. Но твоя походка позовет меня, точно музыка, и я выйду из своего убежища.

2. Отсутствие; Между; CTL.

3. Взаимное исключение Пнуэли

Для двух процессов существует одна общая переменная s, которая равна либо 0, либо 1, и первоначально её значение равно 1. Кроме того, каждый процесс имеет локальную логическую переменную y, которая первоначально равна 0. Процесс P_i (i = 0, 1) задан следующим образом:

```
loop forever do
begin
     Noncritical section
     (y_i, s) := (1, i);  // атомарный шаг
     wait until ((y_1-i = 0) ∨ (s ≠ i));
     Critical section
     y_i := 0
end
```

ЗАДАНИЕ 10

1. **Алиса в Стране чудес**

   Если я и дальше буду так уменьшаться, - сказала она про себя, - я могу и вовсе исчезнуть.

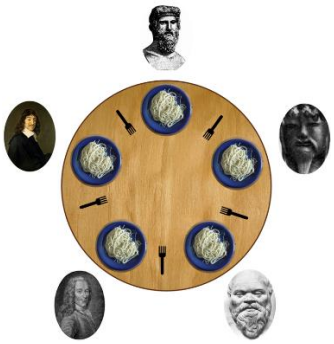
2. Универсальность; Перед; CTL.

3. Обедающие философы (Дейкстра)

Трое молчаливых философов сидят вокруг круглого стола, перед каждым философом стоит тарелка спагетти. Вилки лежат на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять. Приём пищи не ограничен количеством оставшихся спагетти — подразумевается бесконечный запас. Тем не менее, философ может есть только тогда, когда держит две вилки — взятую справа и слева.

Каждый философ может взять ближайшую вилку (если она доступна), или положить — если он уже держит её. Взятие каждой вилки и возвращение её на стол являются раздельными действиями, которые должны выполняться одно за другим.



ЗАДАНИЕ 11

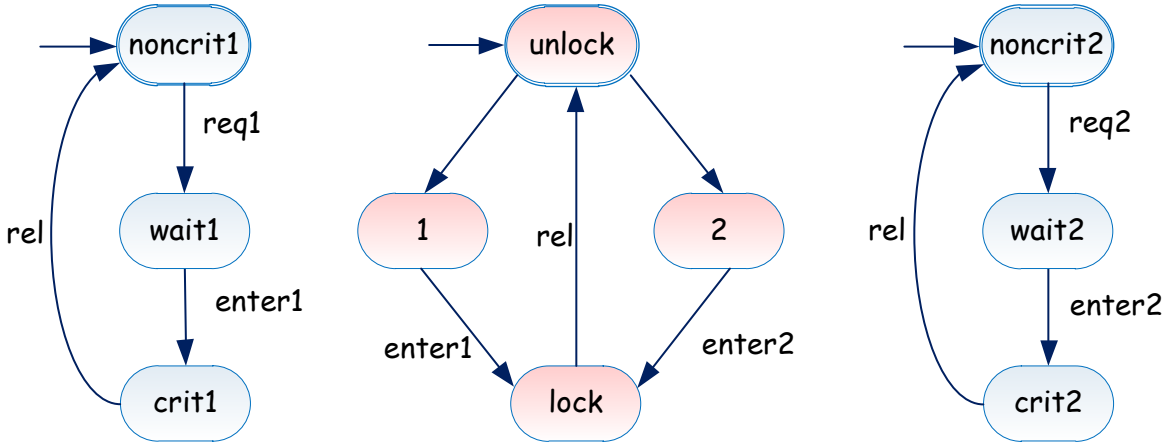
1. **Алиса в Стране чудес**

   - Ах, боже мой, что скажет Герцогиня! Если я опоздаю, она выйдет из себя и не вернётся обратно, а прямо придёт в ярость!

2. Отсутствие; Перед; LTL.

3. Взаимное исключение с рандомизированным арбитром

В системе взаимного исключения двух процессов рандомизированный арбитр решает, какой процесс получает доступ к критической секции. Он делает это, бросая монеты. Мы абстрагируемся от вероятностей и моделируем бросок монеты путем недетерминированного выбора между альтернативами «1» и «2». Предполагается, что два конкурирующих процесса взаимодействуют с арбитром посредством синхронизации действий *enter1* и *enter2*. Критическая секция освобождается путем синхронизации над относительно действия *rel*. Для простоты мы воздерживаемся от указания того, какой процесс освобождает критический участок.



ЗАДАНИЕ 12

1. **Алиса в Стране чудес**

   Ты бы еще могла стучать, - продолжал Лягушонок, не отвечая на вопрос,- если б между нами была дверь. Например, если б ты была _там_, ты бы постучала, и я бы тогда тебя выпустил.

2. Отсутствие; После; LTL.

3. Система управления лифтом.

Лифт можно вызвать с любого этажа вниз или вверх. Лифт останавливается на этажах с вызовом и подбирает там людей, если количество человек не превышает максимально допустимое.

ЗАДАНИЕ 13

1. **Служебный роман**

   Если сегодня еще кто-нибудь умрет или родится, то я останусь без обеда.

2. Предшествование; Постоянно; LTL.

3. Алгоритм выбора лидера Питерсона

Пусть топология коммуникационных FIFO-каналов, связывающих 5 процессов, является кольцом. Процесс может отправлять сообщения только по часовой стрелке. Первоначально каждый процесс имеет уникальный идентификатор *ident* (натуральное число). Процесс может быть активным или эстафетным. Сначала процесс активен. В результате работы процессов по следующему протоколу должен быть избран лидер (процесс с максимальным номером):

```
active:
d := ident;
do forever
begin
      /* start phase */
      send(d);
      receive(e);
      if e = ident then announce elected;
      if d > e then send(d) else send(e);
      receive(f);
      if f = ident then announce elected;
      if e ≥ max(d, f) then d := e else goto relay;
end
relay:
do forever
begin
      receive(d);
      if d = ident then announce elected;
      send(d)
end
```

ЗАДАНИЕ 14

1. **Симпсоны**

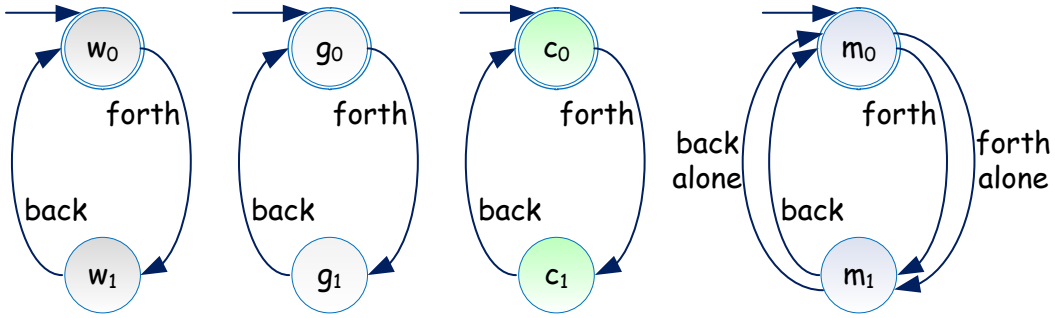
   — Что ты скажешь, если я пойду работать в тюрьму? — Сначала я захочу сэндвич. А потом буду тобой гордиться.

2. Существование; Постоянно; LTL.

3. Волк, коза и капуста.

Однажды крестьянину понадобилось перевезти через реку волка, козу и капусту. У крестьянина есть лодка, в которой может поместиться, кроме самого крестьянина, только один объект — или волк, или коза, или капуста. Если крестьянин оставит без присмотра волка с козой, то волк съест козу; если крестьянин оставит без присмотра козу с капустой, коза съест капусту.

Как крестьянину перевезти на другой берег всё своё имущество в целости и сохранности?



ЗАДАНИЕ 15

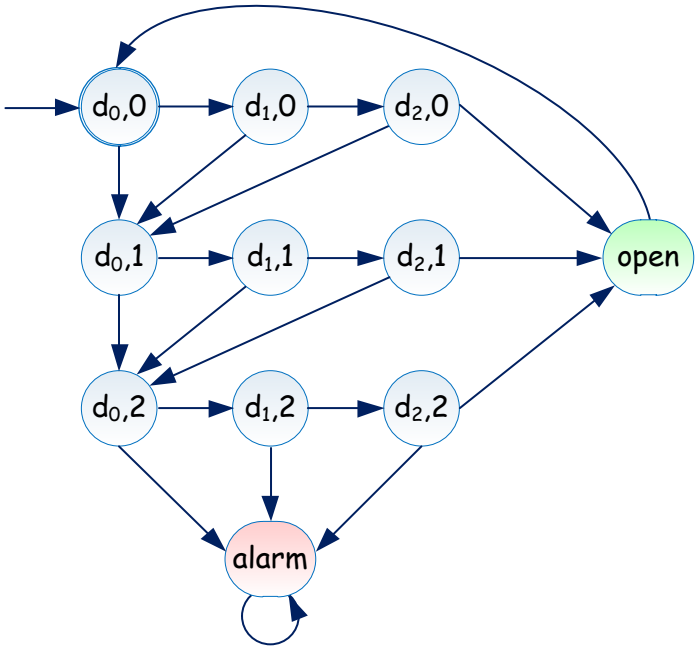
1. **Симпсоны**

Чем раньше дети заговорят, тем чаще будут перечить.

2. Ответ; Постоянно; LTL.

3. Кодовый замок

Устройство открывания двери требует трехзначного кода $d_0d_1d_2$, где $d_i \in \{0, \dots, 9\}$. Можно ввести ошибочную цифру, но не чаще двух раз. Переменная *error* отслеживает количество неправильных цифр, которые были введены, и сначала её значение равно нулю. В случае, если значение *error* превышает два, устройство открывания двери выдает сигнал тревоги. При успешном вводе кода двери дверь открывается. После этого дверь снова блокируется и система возвращается в исходное состояние.



ЗАДАНИЕ 16

1. **Симпсоны**

   Но потом мы узнали закон шоу-бизнеса: всё, что возвышается, потом падает.

2. Отсутствие; Постоянно; LTL.

3. Производитель-потребитель

В параллельной системе есть процесс производителя и потребителя, которые используют один буфер элементов данных ёмкости *m>0*. Производитель последовательно производит *n* элементов и вставляет эти элементы данных один за другим в буфер. Локальная переменная *in* указывает следующую ячейку буфера, которая должна быть записана. Производитель может вставить элемент в буфере ячейки по индексу, только когда эта ячейка пуста (⊥). Процесс-потребитель может последовательно получать элементы данных из буфера и потом потребить их всех разом.

```
Producer
      in := 0;
      while true {
            produce d1, … , dn;
            for i = 1 to n {
                  wait until (buffer[in] = ⊥) {
                        buffer[in] := di;
                        in := (in + 1) mod m; }
            }
      }

Consumer
      out := 0;
      while true {
            for j = 1 to n {
                  wait until (buffer[out]    = ⊥) {
                        ej := buffer[out];
                        buffer[out] := ⊥;
                        out := (out + 1) mod m; }
            }
            consume e1, … , en
      }
```

ЗАДАНИЕ 17

1. **Симпсоны**

— Мы идём на родительское собрание, вернёмся к ужину.— А что будет на ужин?— Смотря что скажут учителя. Если вы хорошо учились, то ждите пиццу, если учились плохо, то... яд.— А если один учился плохо, а другой — хорошо? Пиццу с ядом?— Нет, две покупки я делать не собираюсь!

2. Универсальность; Постоянно; LTL.

3. Проблема спящего парикмахера (Дейкстра)

В парикмахерской работает один парикмахер. У парикмахера есть одно рабочее место и приемная с несколькими стульями. Когда парикмахер заканчивает подстригать клиента, он отпускает клиента и затем идет в приёмную, чтобы посмотреть, есть ли там ожидающие клиенты. Если они есть, он приглашает одного из них и стрижет его. Если ждущих клиентов нет, он возвращается к своему креслу и спит в нем.

Каждый приходящий клиент смотрит на то, что делает парикмахер. Если парикмахер спит, то клиент будит его и садится в кресло. Если парикмахер работает, то клиент идет в приёмную. Если в приёмной есть свободный стул, клиент садится и ждёт своей очереди. Если свободного стула нет, то клиент уходит.

ЗАДАНИЕ 18

1. **Симпсоны**

   Не стоит горевать. Люди постоянно умирают. Как знать, может и ты завтра проснешься мертвым.

2. Универсальность; После; LTL.

3. Задача о курильщиках (Патил)

Три заядлых курильщика сидят за столом. Каждому из них доступно бесконечное количество одного из трёх компонентов: у одного курильщика — табака, у второго — бумаги, у третьего — спичек. Для того чтобы делать и курить сигары, необходимы все три компонента.

Также, кроме курильщиков, есть бармен, помогающий им делать сигареты: он недетерминированно выбирает двух курильщиков, берёт у них по одному компоненту из их запасов и кладёт их на стол. Третий курильщик забирает ингредиенты со стола и использует их для изготовления сигареты, которую он курит некоторое время. В это время бармен, завидев стол пустым, снова выбирает двух курильщиков случайным образом и кладёт их компоненты на стол. Процесс повторяется бесконечно.

Курильщики, по условию проблемы, честные: они не прячут компоненты, выданные барменом, — они лишь скручивают сигарету тогда, когда докурят предыдущую. Если бармен кладёт, например, табак и бумагу на стол, пока поставщик спичек курит, то табак и бумага останутся нетронутыми на столе, пока курильщик со спичками не докурит сигарету и только затем не возьмёт табак и бумагу.

ЗАДАНИЕ 19

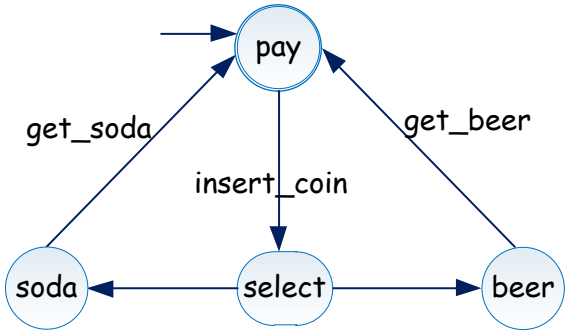
1. **Даниил Хармс**

Но если как-нибудь его
Случится встретить вам,
Тогда скорей,
Тогда скорей,
Скорей скажите нам.

2. Предшествование; После-пока; CTL.

3. Автомат для выдачи напитков

Система переходов на рисунке моделирует автомат для выдачи напитков. Пользователь вставляет монетиу, после этого автомат недетерминированно выбирает выдаёт пиво или содовую и затем выдаёт выбранный напиток.



ЗАДАНИЕ 20

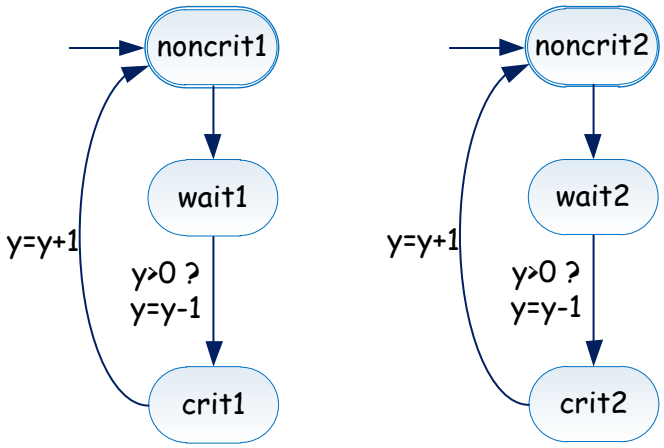
1. **Книга Екклезиаста**

   Что было, то и будет; и что делалось, то и будет делаться, и нет ничего нового под солнцем.

2. Предшествование; После; CTL.

3. Взаимное исключение с помощью семафоров

Процессы P_1 и P_2 представлены системами переходов на рисунке. Они совместно используют двоичный семафор y. Когда y = 0 это указывает на то, что семафор - блокировка для доступа к критической секции - в настоящее время занят одним из процессов. Когда y = 1, семафор свободен. Система переходов процессов представлены на рисунке. Для простоты локальные переменные и общие переменные, отличные от y, не рассматриваются. Кроме того, действия внутри и снаружи критических секций опущены. Состояния $noncrit_i$ представляют некритические действия, $wait_i$ моделируют ситуацию, в которой P_i ждет входа в критическую секцию, и $crit_i$ моделируют пребывание в критической секции.



ЗАДАНИЕ 21

1. **Евангелие от Матфея**

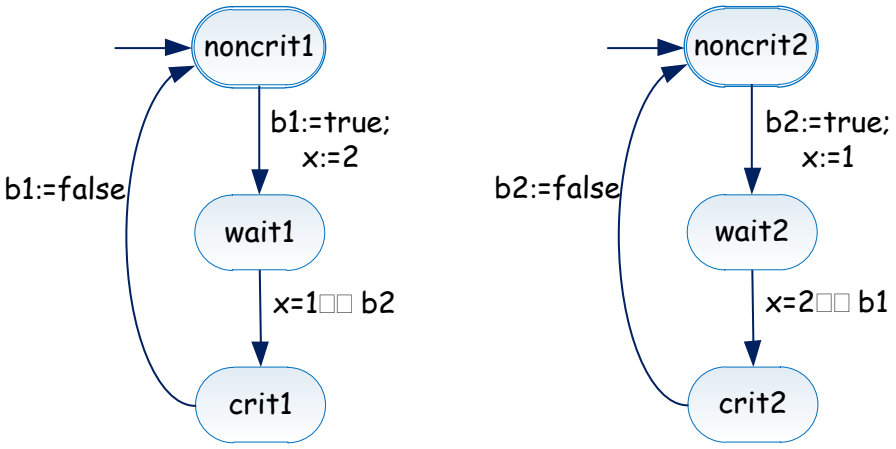
   Многие же будут первые последними, и последние первыми.

2. Ответ; После; CTL.

3. Взаимное исключение Питерсона

Рассмотрим процессы P_1 и P_2 с общими переменными b_1, b_2 и x. b_1 и b_2 являются булевыми переменными, а x∈{1, 2}. Стратегия планирования реализуется с использованием x следующим образом. Если оба процесса хотят войти в критическую секцию (т. е. находятся в состоянии $wait_i$), значение переменной x определяет, какой из двух процессов может войти в его критическую секцию: если $x = i$, то P_i может войти в свою критическую секцию (для i = 1, 2). При входе в позицию $wait_1$ процесс P_1 выполняет x: = 2, тем самым давая право на обработку P_2 для входа в критическую секцию. Таким образом, значение x указывает, чья очередь входа в критическую секцию. Симметрично, P_2 устанавливает x в 1, когда начинает ждать. Переменные b_i предоставляют информацию о текущем местоположении P_i. Точнее, $b_i = wait_i$ ∨ criti. b_i устанавливается, когда P_i начинает ждать. В псевдокоде выполнения P_1 выглядит следующим образом (для процесса P_2 аналогично):

```
P1 loop forever
… (* noncritical actions *)
b1 := true; x := 2; (* request *)
wait until (x = 1 ∨ ¬b2)
do critical section od
b1 := false (* release *)
... (* noncritical actions *)
end loop
```



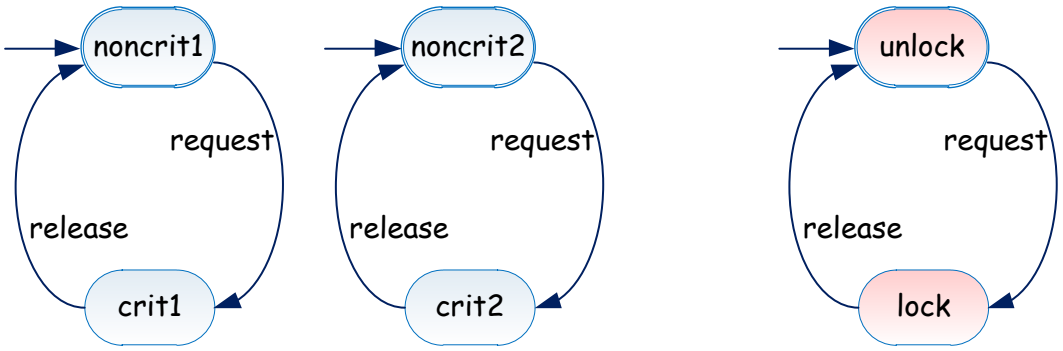
ЗАДАНИЕ 22

1. **Книга Бытия**
   - **а.** *…в поте лица твоего будешь есть хлеб, доколе не возвратишься в землю, из которой ты взят, ибо прах ты и в прах возвратишься.*

2. Предшествование; Перед; CTL.

3. Взаимное исключение с арбитром

Взаимное исключение между процессами P_1 и P_2 заключается в моделировании двоичного семафора, который регулирует доступ к критической секции посредством отдельного параллельного процесса *Arbiter*, который взаимодействует с P_1 и P_2 посредством хэндшейкинга для синхронизации действий. Для простоты мы игнорируем фазу ожидания и предполагаем, что P_i чередуется бесконечно часто между некритическими и критическими секциями. Упрощённые системы переходов процессов имеют только два состояния: $crit_i$ и $noncrit_i$. Процесс *Arbiter* имитирует двоичный семафор. P_1 и P_2 общаются с *Arbiter* для синхронизации действий *request* и *release*.



ЗАДАНИЕ 23

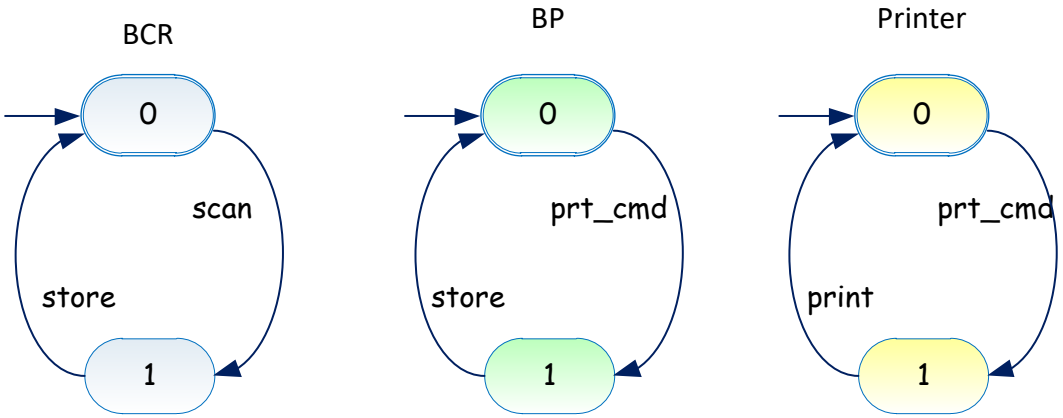
1. **Хагакурэ**

   Всю свою жизнь прилежно учись. Каждый день становись более искусным, чем ты был за день до этого, а на следующий день – более искусным, чем сегодня.

2. Существование; После; CTL.

3. Система бронирования.

Систему бронирования на кассе супермаркета состоит из трех процессов: считывателя штрих-кода *BCR*, фактической программы бронирования *BP* и принтера *Printer*. Считыватель штрих-кода *BCR* считывает штрих-код и передает данные только что отсканированного продукта в программу бронирования *BP*. При получении данных программа бронирования *BP* передает цену товара на принтер *Printer*, который печатает идентификатор товара и цену на чек. Синхронное взаимодействие между считывателем штрих-кода *BCR* и программой бронирования *BP*, а также между программой бронирования *BP* и принтером *Printer* выполняется посредством хэндшейкинга. Каждый процесс находится в одном из двух состояний 0 и 1.



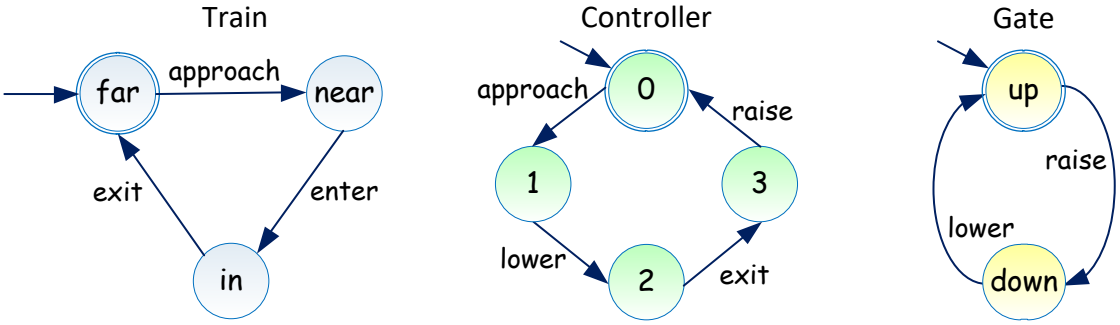
ЗАДАНИЕ 24

1. **«Свет в августе» (1932) — роман американского писателя Уильяма Фолкнера.**
   Память верит раньше, чем вспоминает знание. Верит дольше, чем помнит, дольше, чем знание спрашивает.

2. Цепное предшествование 1-2; Постоянно; CTL.

3. Железнодорожный переезд

Для железнодорожного переезда необходимо разработать систему управления, которая по получении сигнала, указывающего, что поезд приближается, закрывает шлагбаум и открывает его только после того, как поезд отправил сигнал, указывающий, что он пересек дорогу. Система состоит из трех компонентов: *Train*, *Gate* и *Controller*.
Для простоты предполагаем, что все поезда проходят соответствующий участок дороги в том же направлении - слева направо. Состояния системы переходов для *Train* имеют следующий интуитивный смысл: в состоянии *far* поезд далеко от переезда, в состоянии *near* он приближается к переходу и только что отправил сигнал, чтобы уведомить об этом, и в состоянии *in* находится на переезде. Интерпретация состояний *Gate* очевидна. *Controller* синхронизирует изменения своих состояний с *Train* (относительно действий *approach* и *exit*) и с *Gate* (относительно действий *lower* и *raise*).



ЗАДАНИЕ 25

1. **Град обреченный**

   Великий стратег стал великим именно потому, что понял (а может быть, знал от рождения): выигрывает вовсе не тот, кто умеет играть по всем правилам; выигрывает тот, кто умеет отказаться в нужный момент от всех правил, навязать игре свои правила, неизвестные противнику, а когда понадобится — отказаться и от них.

2. Универсальность; После-пока; CTL.

3. Протокол альтернирующего бита (ABP)
Простой протокол передачи данных, который повторно передает потерянные или поврежденные сообщения.

Сообщения отправляются от передатчика A к приемнику B. Предположим, что канал от A до B инициализирован и что нет сообщений в пути. Каждое сообщение содержит часть данных, контрольную сумму и однобитовый порядковый номер, то есть значение, которое равно 0 или 1.

Когда A отправляет сообщение, он отправляет его непрерывно с тем же порядковым номером, пока он не получит подтверждение подтверждения *ACK* из B, которое содержит тот же порядковый номер. Когда это произойдет, A дополняет (переворачивает) порядковый номер и начинает передачу следующего сообщения.

Когда B получает сообщение от A, он проверяет контрольную сумму. Если сообщение не повреждено, B отправляет обратно *ACK* с тем же порядковым номером. Если это первое сообщение с этим порядковым номером, оно отправляется для обработки. Последующие сообщения с одним и тем же битом последовательности просто подтверждаются. Если сообщение повреждено, B отправляет обратно отрицательное подтверждение *NAK*. Это необязательно, так как A продолжит передачу, пока не получит правильный *ACK*.

A обрабатывает поврежденные сообщения *ACK* и сообщения *NAK* таким же образом. Самое простое поведение - игнорировать их все и продолжить передачу.

ЗАДАНИЕ 26

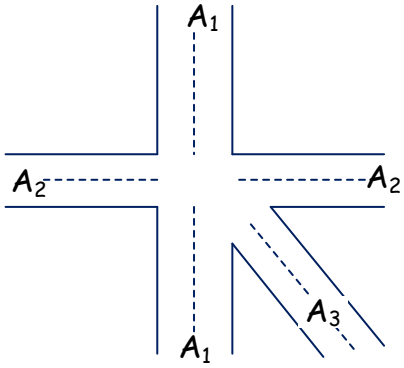
1. **Маленький принц**

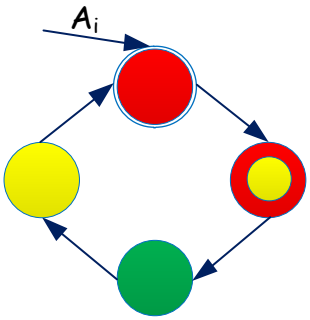
   Довольно только передвинуть стул на несколько шагов. И ты снова и снова смотришь на закатное небо, стоит только захотеть ….

2. Существование; Перед; CTL.

3. Светофор.

Построить систему светофоров, которые включают зеленый свет в следующем порядке: A1, A2, A3, A1, A2, A3, ….





ЗАДАНИЕ 27

1. **Над пропастью во ржи**

   Еще будут приходить по воскресеньям, класть тебе цветы на живот.

2. Цепной ответ 1-2; Постоянно; CTL.

3. Взаимное исключение Пнуэли

Для двух процессов существует одна общая переменная s, которая равна либо 0, либо 1, и первоначально её значение равно 1. Кроме того, каждый процесс имеет локальную логическую переменную y, которая первоначально равна 0. Процесс P_i (i = 0, 1) задан следующим образом:

```
loop forever do
begin
     Noncritical section
     (y_i, s) := (1, i);  // атомарный шаг
     wait until ((y_{1-i} = 0) ∨ (s ≠ i));
     Critical section
     y_i := 0
end
```

ЗАДАНИЕ 28

1. **Алиса в Стране чудес**

   *- Что ж, - сказала Алиса, - я так и сделаю (*съем пирожок – прим.*). Если при этом я вырасту, я достану ключик, а если уменьшусь - пролезу под дверь.*

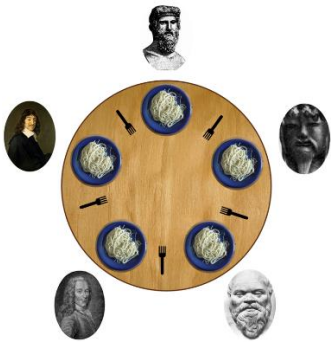
2. Ответ; После-пока; CTL.

3. Обедающие философы (Дейкстра)

Трое молчаливых философов сидят вокруг круглого стола, перед каждым философом стоит тарелка спагетти. Вилки лежат на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять. Приём пищи не ограничен количеством оставшихся спагетти — подразумевается бесконечный запас. Тем не менее, философ может есть только тогда, когда держит две вилки — взятую справа и слева.

Каждый философ может взять ближайшую вилку (если она доступна), или положить — если он уже держит её. Взятие каждой вилки и возвращение её на стол являются раздельными действиями, которые должны выполняться одно за другим.



ЗАДАНИЕ 29

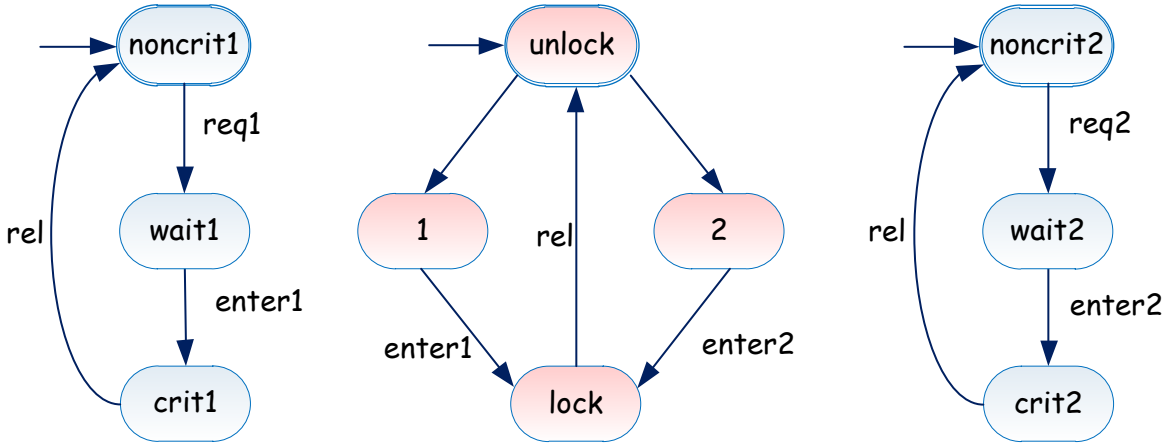
1. **Алиса в Стране чудес**

   По крайней мере _здесь_ мне расти больше некуда.- А вдруг я на этом и остановлюсь? - думала Алиса. - Пожалуй, это неплохо - я тогда не состарюсь! Правда, мне придется всю жизнь учить уроки.

2. Отсутствие; Перед; CTL.

3. Взаимное исключение с рандомизированным арбитром

В системе взаимного исключения двух процессов рандомизированный арбитр решает, какой процесс получает доступ к критической секции. Он делает это, бросая монеты. Мы абстрагируемся от вероятностей и моделируем бросок монеты путем недетерминированного выбора между альтернативами «1» и «2». Предполагается, что два конкурирующих процесса взаимодействуют с арбитром посредством синхронизации действий *enter1* и *enter2*. Критическая секция освобождается путем синхронизации над относительно действия *rel*. Для простоты мы воздерживаемся от указания того, какой процесс освобождает критический участок.



ЗАДАНИЕ 30

1. **Вино из одуванчиков**

   Как бы ты ни старалась оставаться прежней, ты все равно будешь только такой, какая ты сейчас, сегодня.

2. Отсутствие; После; CTL.

3. Система управления лифтом.

Лифт можно вызвать с любого этажа вниз или вверх. Лифт останавливается на этажах с вызовом и подбирает там людей, если количество человек не превышает максимально допустимое.

ЗАДАНИЕ 31

1. **Ирония судьбы, или С лёгким паром!**

   Да, я хирург. Мне часто приходится делать людям больно, чтобы потом им жилось хорошо.

2. Предшествование; Постоянно; CTL.

3. Алгоритм выбора лидера Питерсона

Пусть топология коммуникационных FIFO-каналов, связывающих 5 процессов, является кольцом. Процесс может отправлять сообщения только по часовой стрелке. Первоначально каждый процесс имеет уникальный идентификатор *ident* (натуральное число). Процесс может быть активным или эстафетным. Сначала процесс активен. В результате работы процессов по следующему протоколу должен быть избран лидер (процесс с максимальным номером):

```
active:
d := ident;
do forever
begin
      /* start phase */
      send(d);
      receive(e);
      if e = ident then announce elected;
      if d > e then send(d) else send(e);
      receive(f);
      if f = ident then announce elected;
      if e ≥ max(d, f) then d := e else goto relay;
end
relay:
do forever
begin
      receive(d);
      if d = ident then announce elected;
      send(d)
end
```

ЗАДАНИЕ 32

1. **Симпсоны**

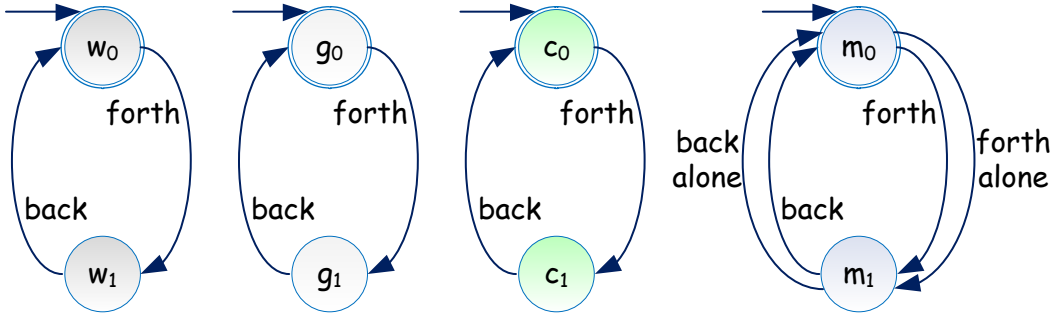
Зачем надевать носки, если через неделю всё равно придётся их снимать?

2. Существование; Постоянно; CTL.

3. Волк, коза и капуста.

Однажды крестьянину понадобилось перевезти через реку волка, козу и капусту. У крестьянина есть лодка, в которой может поместиться, кроме самого крестьянина, только один объект — или волк, или коза, или капуста. Если крестьянин оставит без присмотра волка с козой, то волк съест козу; если крестьянин оставит без присмотра козу с капустой, коза съест капусту.

Как крестьянину перевезти на другой берег всё своё имущество в целости и сохранности?



ЗАДАНИЕ 33

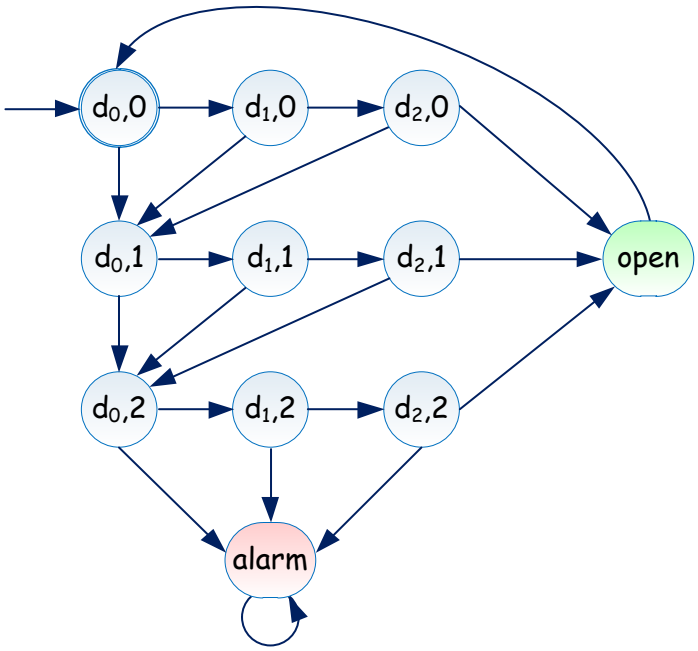
1. **Симпсоны**

   Мэгги держит плакат с надписью: «Остановите меня до того, как я снова попытаюсь убить своего отца». На плакате Гомера написано: «Запретите жестокие мультфильмы. В следующий раз мне вряд ли повезёт».

2. Ответ; Постоянно; CTL.

3. Кодовый замок

Устройство открывания двери требует трехзначного кода $d_0d_1d_2$, где $d_i \in \{0, \dots, 9\}$. Можно ввести ошибочную цифру, но не чаще двух раз. Переменная *error* отслеживает количество неправильных цифр, которые были введены, и сначала её значение равно нулю. В случае, если значение *error* превышает два, устройство открывания двери выдает сигнал тревоги. При успешном вводе кода двери дверь открывается. После этого дверь снова блокируется и система возвращается в исходное состояние.



ЗАДАНИЕ 34

1. **Симпсоны**

   — Все, записку я отправил. Захочет твоя семья увидеть тебя живым — заплатит. — Ну не знаю... Они меня сто лет живым видели бесплатно...

2. Отсутствие; Постоянно; CTL.

3. Производитель-потребитель

В параллельной системе есть процесс производителя и потребителя, которые используют один буфер элементов данных ёмкости *m>0*. Производитель последовательно производит *n* элементов и вставляет эти элементы данных один за другим в буфер. Локальная переменная *in* указывает следующую ячейку буфера, которая должна быть записана. Производитель может вставить элемент в буфере ячейки по индексу, только когда эта ячейка пуста (⊥). Процесс-потребитель может последовательно получать элементы данных из буфера и потом потребить их всех разом.

```
Producer
      in := 0;
      while true {
            produce d1, … , dn;
            for i = 1 to n {
                  wait until (buffer[in] = ⊥) {
                        buffer[in] := di;
                        in := (in + 1) mod m; }
            }
      }

Consumer
      out := 0;
      while true {
            for j = 1 to n {
                  wait until (buffer[out]    = ⊥) {
                        ej := buffer[out];
                        buffer[out] := ⊥;
                        out := (out + 1) mod m; }
            }
            consume e1, … , en
      }
```

ЗАДАНИЕ 35

1. **Симпсоны**

   Не вижу смысла выходить из дома. Мы все равно каждый раз возвращаемся обратно.

2. Универсальность; Постоянно; CTL.

3. Проблема спящего парикмахера (Дейкстра)

В парикмахерской работает один парикмахер. У парикмахера есть одно рабочее место и приемная с несколькими стульями. Когда парикмахер заканчивает подстригать клиента, он отпускает клиента и затем идет в приёмную, чтобы посмотреть, есть ли там ожидающие клиенты. Если они есть, он приглашает одного из них и стрижет его. Если ждущих клиентов нет, он возвращается к своему креслу и спит в нем.

Каждый приходящий клиент смотрит на то, что делает парикмахер. Если парикмахер спит, то клиент будит его и садится в кресло. Если парикмахер работает, то клиент идет в приёмную. Если в приёмной есть свободный стул, клиент садится и ждёт своей очереди. Если свободного стула нет, то клиент уходит.

ЗАДАНИЕ 36

1. **Футурама**

   Ты не можешь до конца жизни лежать в кровати, как овощ. Я пробовал, пролежни болят.

2. Универсальность; После; CTL.

3. Задача о курильщиках (Патил)

Три заядлых курильщика сидят за столом. Каждому из них доступно бесконечное количество одного из трёх компонентов: у одного курильщика — табака, у второго — бумаги, у третьего — спичек. Для того чтобы делать и курить сигары, необходимы все три компонента.

Также, кроме курильщиков, есть бармен, помогающий им делать сигареты: он недетерминированно выбирает двух курильщиков, берёт у них по одному компоненту из их запасов и кладёт их на стол. Третий курильщик забирает ингредиенты со стола и использует их для изготовления сигареты, которую он курит некоторое время. В это время бармен, завидев стол пустым, снова выбирает двух курильщиков случайным образом и кладёт их компоненты на стол. Процесс повторяется бесконечно.

Курильщики, по условию проблемы, честные: они не прячут компоненты, выданные барменом, — они лишь скручивают сигарету тогда, когда докурят предыдущую. Если бармен кладёт, например, табак и бумагу на стол, пока поставщик спичек курит, то табак и бумага останутся нетронутыми на столе, пока курильщик со спичками не докурит сигарету и только затем не возьмёт табак и бумагу.