**Оглавление**

1. Задание
2. Описание процесса
3. Построение метамодели «асинхронный процесс»
4. Свойства процесса

4.1 Эффективность

4.2 Управляемость

4.3 Простота

1. Операции над процессом

5.1 Репозиция

5.2 Редукция

5.3 Композиция

1. Построение сети Петри
2. Анализ свойств мест с. Петри на ограниченность и безопасность
3. Анализ свойств переходов с. Петри на живость и устойчивость
4. Заключение

Список использованной литературы

**1. Задание**

Целью расчетно-графического задания является получение опыта в конструировании метамодели «асинхронный процесс» и модели «сеть Петри» и в исследовании их свойств.

Предлагается выполнить следующее:

1. Выделить компоненты рассматриваемого процесса.
2. Сформировать множество ситуаций рассматриваемого процесса.
3. Описать модель «асинхронный процесс».
4. Определить траектории выполнения процесса и классы эквивалентности ситуаций и сделать вывод о свойствах рассматриваемого процесса (эффективность, управляемость, простота).
5. Определить множество дополнительных ситуаций для возобновления процесса (если они есть) и построить полную или частичную репозицию процесса.
6. Выделить входные или выходные компоненты асинхронного процесса, выбрать требуемые и построить на их основе редукцию процесса.
7. Определить два подпроцесса на базе исследуемого, выбрать удобный вид композиции (последовательную или параллельную) и построить ее.
8. Описать составляющие модели «асинхронный процесс», используя понятия модели «сеть Петри».
9. Провести анализ свойств мест сети Петри на ограниченность и безопасность.
10. Провести анализ свойств переходов сети Петри на живость и устойчивость.

**2. Описание процесса**

асинхронный репозиция редукция сеть пери

В качестве вычислительного процесса был выбран процесс со следующим названием: «Запись информации на магнитный носитель». Кратко опишем его суть.

Процесс записи на магнитный диск является достаточно сложной задачей и составить метамодель «асинхронный процесс» во всех деталях соответствующую реальному процессу не представляется возможным, так как неизбежны сильная детализация, а так же повышение сложности модели.

Процесс записи опишем следующей последовательностью команд и операций: магнитный диск вставлен в дисковод и готов для записи (часть магнитной поверхности уже может содержать какую-либо информацию). Дисковод состоит из множества механических деталей и механизмов, управление которыми осуществляет микроконтроллер – устройство, которое получает команды от центрального процессора и выполняет их.

Запись информации производится специальной записывающей головкой, которая подводится к соответствующей дорожке и сектору с помощью системы позиционирования головки. Запись информации происходит посекторно. Количество секторов для записи содержится в специальном регистре контроллера и декрементируется после записи очередного сектора.

После того как очередной сектор записан, головка позиционируется на следующий сектор. Очередная порция данных (для записи) считывается по шине данных.

После того, как данные записаны, производится контроль записи (верификация или CRC (проверка контрольной суммы)). Такая операция необходима для проверки правильности записанной информации. В случае ошибки записи, специальный регистр (Data Error Register) устанавливается в единицу и запись прекращается. Головки при этом отводятся от поверхности.

Успешная запись всех секторов завершается сбрасыванием в 0 регистра ошибок данных. Регистр, содержащий количество секторов для записи так же обнуляется. Головки отводятся от поверхности магнитного носителя и контроллер выдает сигнал по шине данных процессору об окончании операции ввода/вывода.

**3. Построение метамодели «асинхронный процесс»**

Прежде всего, выделим компоненты рассматриваемого процесса:

|  |  |  |
| --- | --- | --- |
| N | Обозначение | Описание |
| 1 | K | Микроконтроллер дисковода. Работает (K+) |
| 2 | S | Свободное место. Есть в наличии (S+) или нет (S-) |
| 3 | E | Регистр, сигнализирующий ошибку. Произошла ошибка (E+) |
| 4 | P | Механизм позиционирования головки записи. Задействован (P+) |
| 5 | W | Головка записи. Записывает информацию (W+) |
| 6 | C | Блок контроля правильности записи (crc check). Проводится проверка (C+) |
| 7 | F | Fat таблица. Обновление fat таблицы после записи (F+) |

Для описания процессов и задания их взаимодействия требуется структурирование ситуаций. Структурируем ситуации по первому способу, при котором каждая из ситуаций представляется двоичным вектором, размерность которого равна числу семантически задаваемых компонент, а число единиц вектора соответствует числу истинных в этой ситуации предикатов.

Используя выделенные компоненты, опишем ситуации, возникающие в процессе записи информации на магнитный носитель:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ситуация | K | S | E | P | W | C | F | Описание |
| S1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Контроллер активен и ожидает команду от процессора на запись |
| S2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Есть свободное место для записи |
| S3 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | Места нет. Регистр ошибки устанавливается в код ошибки. |
| S4 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | Головка подводится на нужный сектор |
| S5 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | Запись информации в сектор |
| S6 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | Проверка валидности записанных данных |
| S7 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | Успешная запись, головки отводятся от поверхности, обновляется таблица fat. |
| S8 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Некорректная запись информации. |

Назовем асинхронным процессом четверку <S,F,I,R>, в которой: S - непустое множество ситуаций;

F - отношение непосредственного следования ситуаций, определенное на множестве S×S (F⊂ S×S);

I - множество инициаторов (I⊂S), т.е. таких ситуаций из S, для которых если iFSk, i∈I, Sk∈I, то из SkFSl следует, что Sl∉I;

R - множество результантов (R⊂S), т.е. таких ситуаций из S, для которых, если rFS, r∈R, то S∈R.

Согласно этому определению, имеем для рассматриваемого процесса:

P=<S,F,I,R>

S={ }

I={},

Где -инициирует запись

-инициирует проверку записанных данных

R={ }, где

-нет места для записи

-успешная запись

-ошибка записи информации

Выпишем все возможные траектории процесса:

 - не хватает места для записи

 - успешный процесс записи

 - ошибка записи

 - неудачная проверка записанных данных

 - удачная проверка записанных данных

**4. Свойства процесса**

Рассмотрим свойства асинхронного процесса.

**4.1 Эффективность**

Пусть задан АП, у которого :

1. для любого s∈S \ R найдется r∈R такой, что sMr;
2. для любого s∈S \ I найдется i∈I такой, что iMs;
3. не найдется ситуации Si и Sj таких, что (Si∉R)&(Sj∉R)&(SiMSj)&(SjMSi).

АП, удовлетворяющий свойствам 1 - 3, будет называться эффективным , т.е. из инициаторов эффективного процесса все траектории ведут в результанты (свойство 1 и 3), и каждая из траекторий, приводящих к результанту, начинается в каком-либо инициаторе (св. 1 и 2). Эффективность АП оставляет место недетерминированности, т.е. возможно, что из некоторого инициатора процесс попадает в разные результанты, но он не содержит ориентированных циклов вне ситуаций, принадлежащих I и R.

**Вывод**: данный процесс не является эффективным (не выполняется свойство 3), так как процесс содержит цикл вне ситуаций, принадлежащих I и R.

**4.2 Управляемость**

Если в эффективном АП каждая допустимая последовательность классов ведет из начального класса в один и только один заключительный класс, то такой процесс называется управляемым. Таким образом, в управляемом АП вводится ограничение на степень недетерминизма: все траектории из любого инициатора ведут в один заключительный класс.

Для определения того факта, является процесс управляемым или нет, разобьем множество ситуаций S на классы эквивалентности.

 ,

где класс - начальный, а классы - заключительные.

**Вывод**: данный процесс не является управляемым, так как из начального класса можно попасть в заключительный класс , а также в класс . То есть не выполняется условие детерминированности.

**4.3 Простота**

Пусть в эффективном АП:

1. для любых i∈I и s∈S из iFs ⇒ s∉I;
2. для любых s∈S и r∈R из sFr ⇒ s∉R

(т.е. из инициатора (результанта) нельзя попасть в другой инициатор (результант) т.е. каждая траектория содержит в точности по одному инициатору и результанту). АП, удовлетворяющий свойствам 1 и 2 будет называться простым.

**Вывод**: процесс не является простым. Траектории

содержат по 2 инициатора (отмечены подчеркиванием)

**5. Операции над процессами**

**5.1 Репозиция**

Репозицией АП задается механизм перехода от результанта к инициаторам

Репозицией АП P = <S,F,I,R> назовем эффективный АП P' = <S', F', I', R'>, такой, что S' ⊆ I∪R∪SD, I' ⊆ R, R' ⊆ I.

Ситуации S' репозиции могут содержать лишь те ситуации из исходного процесса, которые являются лишь инициаторами или результантами, и, кроме того, некоторые дополнительные ситуации из SD, отсутствующие в описании исходного АП.

Отношение F' задает траектории переходов от элементов из I' ⊆ R к элементам R' ⊆ I, возможно через дополнительные ситуации из SD.Если I' = R, R' = I, то репозицию назовем полной. Если F' = ∅ , то репозиция не существует, в остальных случаях она называется частичной.

Для рассматриваемого процесса имеем:

P' = <S', F', I', R'>

S' ⊆ I ∪ R ∪ SD = { }SD = ∅

I'={ }R'={ }

F':(повторить процесс записи сначала)

(повторить проверку записи данных)

Траектории переходов, которые задает отношение F' показаны на рисунке пунктирной стрелкой.

**Вывод:** таким образом, построена частичная репозиция асинхронного процесса, суть которой – получение механизма его возобновления. Семантически репозиция означает повтор операции записи (при успешной записи) и повтор операции проверки записанных данных (в случае возникновения ошибки записи).

**5.2 Редукция**

Операция редукции состоит в сведении данного АП к более простому. Такая операция необходима тогда, когда из полного описания процесса хочется выделить некоторую его часть, рассмотрение которой интересно по тем или иным причинам.

Пусть задан неприведенный АП Р = <S,F,I,R>, ситуации которого структурированы по 2-му способу. Образуем р-блочное разбиение множества S процесса Р, в ситуациях каждого блока которого входная компонента принимает фиксированное значение xj, 1jp.

Выберем r<p различных значений входной компоненты (составляющих множество X\*⊂ X). Ситуации, входящие в те блоки разбиения, которые соответствуют выбранным значениям входной компоненты составляют подмножество S\*, S\*⊂ S.

Для каждого инициатора si∈I постоим множество ситуаций S(si), встречающихся на траекториях процесса Р, ведущих из указанного инициатора.

Образуем множество S(X\*), как объединение тех множеств S(si), для которых справедливо

S(si)S\*, т.е. S(X\*) =

Построим также F(X\*) = F ∩ (S(X\*)×S(X\*)),I(X\*) = I ∩ S(X\*), R(X\*) = R ∩ S(X\*).

Назовем процесс P(X\*) = <S(X\*), F(X\*), I(X\*), R(X\*)> редукцией неприведенного процесса P = <S,F,I,R> по выбранному множеству Х\* значений входной компоненты.

Для рассматриваемого процесса имеем:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ситуация | K | S | E | P | W | C | F |
| S1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| S2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| S3 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| S4 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| S5 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| S6 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| S7 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| S8 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

S={ }

Выберем в качестве значений входной компоненты первые три элемента вектора.

Выпишем множество X={ 100, 110, 101, 111 }

Редукцию сделаем по следующему множеству X\*={100, 101}, то есть семантически рассмотрим ситуацию, когда нет свободного места для записи.

Тогда S\* = { }.

Рассмотрим траектории процесса:

 (подходит, так как ситуации принадлежат S\*)

 (не подходит, т.к. не принадлежат S\*)

 (не подходит)

 (не подходит)

 (не подходит)

Образуем множество S(X\*)={ }, I(X\*)={},R(X\*)= {}

F(X\*):

**Вывод:** построив редукцию данного процесса по выбранным значениям входной компоненты, мы упростили процесс и рассмотрели только ту часть его, в которой складывается ситуация нехватки места на диске. В итоге получаем одну ветвь процесса, в которой фиксируется ошибка и код ее заносится в регистр ошибок.

**5.3 Композиция**

Обозначим исходный процесс как P2. Для него имеем:

S2={ }I2={}R2={ }

Рассмотрим новый процесс P1, состоящий из следующих компонент:

|  |  |  |
| --- | --- | --- |
| N | Обозначение | Описание |
| 1 | K | Контроллер дисковода |
| 2 | P | Дисковод работает. Есть электропитание (P+) |
| 3 | D | Диск вставлен в дисковод (D+) |

|  |  |  |  |
| --- | --- | --- | --- |
| Ситуация | K | PW | D |
| S1 | 1 | 1 | 0 |
| S2 | 1 | 1 | 1 |

Для процесса P1 имеем: S1={ }I1={ }R1={ }

Построим последовательную композицию процессов P1 и P2.

Х\*={1}

Построим редукцию процесса P2 по X\*.

S2\*={ }

Траектории:

(подходит)

(подходит)

(подходит)

(подходит)

(подходит)

Следовательно: S2 (X\*) = { } I2 (X\*) = { } R2 (X\*)= { }

Далее построим редукцию процесса P1 по X\*.

S1\*={ }

Траектории:

Следовательно: S1(X\*) = { } I1 (X\*) = { } R1 (X\*)= { }

Теперь построим процесс P3.

Для процесса P3 имеем: S3={ I3={ }R3={ }



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ситуация | K | S | E | P | W | C | F | PW | D |
| S1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| S2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| S3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| S4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| S5 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| S6 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| S7 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| S8 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| S9 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| S10 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

**Вывод:** таким образом построена последовательная композиция процессов P1 и P2 . Семантически ситуации процесса P1 предшествуют ситуациям процесса P2. Их суть – процесс подготовки носителя к записи информации.

**6. Построение сети Петри**

Сетью Петри называется пятерка N = <P, T, M0, H, F>, где

Р = {p1,...,pn} - конечное непустое множество условий

T = {t1,...,tm} - конечное непустое множество событий

 - функция инцидентности

М0 : Р → {0, 1, 2,...} - начальная разметка.

Сеть Петри есть модельная интерпретация АП.Ситуациями в сети является начальная разметка М0 и все разметки, достижимые от М0, т.е. М∈R(N). Отношение F для любой возможной разметки М задает все разметки, которые могут непосредственно следовать за М. Очевидно, что на множестве R(N) можно определить отношение эквивалентности разметок и задать отношение F непосредственного следования для классов эквивалентности.

Построим сеть Петри для следующей траектории рассматриваемого процесса:

, что семантически соответствует успешному процессу записи данных.

Выпишем значения компонент. Компонента E во всех ситуациях равна 0, поэтому эта компонента не учитывается при построении сети Петри.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | K | S | W | P | C | F |
|  | 1 | 0 | 0 | 0 | 0 | 0 |
|  | 1 | 1 | 0 | 0 | 0 | 0 |
|  | 1 | 1 | 1 | 0 | 0 | 0 |
|  | 1 | 1 | 0 | 1 | 0 | 0 |
|  | 1 | 1 | 0 | 0 | 1 | 0 |
|  | 1 | 1 | 0 | 0 | 0 | 1 |

P1

P2

P6

P5

P4

P3

t1

t2

t3

t5

t4

Формальное описание:

N=<>

**7. Анализ свойств мест с. Петри на ограниченность и безопасность**

Место (условие) р в сети N = (P,T,F,W,M0) называется ограниченным, если существует число n такое, что для любой достижимой в сети разметки М справедливо неравенство М(р) n. Сеть называется ограниченной, если любое ее место ограниченно.

Множество достижимых разметок R(N) конечно, если и только если N - ограниченная сеть.

Место р называется безопасным, если для любого М∈ R(N): М(р) 1, соответственно сеть безопасна, если все ее места безопасны. Любая достижимая в безопасной сети разметка представляет собой вектор из 0 в 1.

Ограниченность и безопасность характеризуют емкость условий: в дискретной информационной системе, моделирующей соотношением систем, можно ограничить емкость накопителей, необходимых для хранения условий наступления событий.

Изобразим граф разметок. (см. рис.).

100000

110000

111000

110100

110010

111001

Разметка M0

**Вывод:**

Все места в сети ограничены, так как для любого места p: M(p)≤n (n=1). Следовательно сеть ограниченная.

Все места являются безопасными, так как для любого места p: M(p) ≤1. Следовательно сеть безопасная (это значит, что все разметки состоят из 0-ей и 1-иц).

**8. Анализ свойств переходов с. Петри на живость и устойчивость**

Переход t в сети Петри N = (P,T,F,W,M0) называется потенциально живым при разметке М∈R(N), если существует M’∈R(N,M) : M’≥ F(p,t), т.е. существует достижимая от М разметка М’, при которой переход t может работать.

Если М = М0, то t называется потенциально живым в сети N.

Переход t – мертвый при М, если он не является потенциально живым при М. Переход t - мертвый, если он мертвый при любой достижимой в сети разметке.

Переход t в сети Петри называется живым, если для любого M∈R(N) cсуществует M’∈ R(N,M) : M ≥ F(p,t), т.е. он потенциально живой при любой достижимой в сети разметке. Сеть называется живой, если все ее переходы живы.

Переход t называется потенциально мертвым, если существует M∈R(N), такая, что при любой разметке M’∈R(N,M) переход t не может работать.

Переход t называется устойчивым в сети N, если t’∈T \ {t}, M∈R(N) : (M ≥ F(p,t)) ∩ (M ≥ F(p,t’))⇒(M ≥ (F(p,t)+F(p,t’))), т.е. если переход t может сработать, то никакой другой переход не может сработав, лишить его этой возможности.

Сеть N устойчива, если все ее переходы устойчивы.

**Вывод:**

Все переходы сети потенциально живы (то есть существует разметка, достижимая от текущей разметки, при которой переход t может сработать), следовательно вся сеть N является живой.

Сеть устойчива, так как все ее переходы являются устойчивыми ( это значит, что если переход t может сработать, то никакой другой переход не может сработав, лишить его этой возможности).

**9. Заключение**

В данном РГЗ была построена модель АП записи на магнитный диск. Для более подробного исследования к данному процессу были применены следующие операции: репозиция, редукция и параллельная композиция. С помощью репозиции мы рассмотрели механизм возобновления процесса нажатия клавиши. С помощью редукции мы упростили процесс и вычленили его отдельную ветвь. Также была построена сеть Петри и изучены ее свойства.

**Список использованной литературы**

1. Конспект лекций по ТВП;

2. А.В. Гордеев, А.Ю. Молчанов, Системное программное обеспечение.