Харьковская государственная академия железнодорожного
транспорта

На правах рукописи

УДК 621.395.345.4: 519.711.2

Журавель Виталий Алексеевич

Разработка метода имитационного моделирования цифровых систем коммутации на основе сетей Петри

Специальность 05.12.02 “Телекоммуникационные системы и управление ими”

Диссертация
на соискание ученной степени кандидата технических наук

Научный руководитель:
Книгавко Николай Владимирович
кандидат технических наук, доцент

г. Харьков 1999

СОДЕРЖАНИЕ.

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

АЛ – абонентская линия (узла коммутации),
ВО – внешнее окружение,
ГСЧ – генератор случайных чисел (функция),
ГСМ – генератор случайных меток (позиция),
ДУК – двухмашинный управляющий комплекс,
ЖГСЧ – ждущий генератор случайных чисел (функция),
ЖГСМ – ждущий генератор случайных меток (позиция),
КС – коммутационная система,
ЛМ – линейный модуль,
МЗС – массив занятости-свободности,
Мо – начальная разметка сети Петри или сетевой модели,
МОВ – массив обслуживаемых вызовов,
МП – массив программ,
МПД – модуль многочастотных приемников и датчиков,
МППК – массив последовательностей периферийных команд,
МСКТ – массив состояния контрольных точек,
ОА – операционный автомат,
ОЗУ – оперативное запоминающее устройство,
ОКЕ – общий канал сигнализации,
ОС – операционная система
ОсЗУ – основное запоминающее устройство,
ПАС – передатчик акустического сигнала,
ПД – покрывающее дерево сети,
ПЗУ – программное постоянное запоминающее устройство,
ПНН – приемник набора номера,
СЛ – соединительная линия (узла коммутации),
СМ – системный модуль,
СП – сеть Петри,
СУВК – специализированный управляющий вычислительный комплекс,
ТМ – терминальный модуль,
УА – управляющий автомат,
УК – узел коммутации,
УС – устройство сопряжения,
УУ – управляющее устройство,
ф. – формула,
ФМ – функциональный модуль,
ЦКП – цифровое коммутационное поле,
ЦПр – центральный процессор,
ЭУС – электронная управляющая система,
SH – множество вершин “Состояние” и “Решение” SDL-диаграммы

ВВЕДЕНИЕ.

Развитие средств распределения информации переживает этап быстрых и радикальных изменений, характеризующийся сращиванием коммутационных систем с системами передачи информации, интегрированием обработки различных видов информации едиными средствами в единой форме, унификацией методов сигнализации, синхронизации, доступа к сети, что получило выражение в концепциях интегральных цифровых сетей стандартов ISDN и BSN, например технология ATM [54]. Велико многообразие таких средств, капиталовложения в них и значимость в жизни общества. В коренных преобразованиях нуждаются существующие общегосударственные и ведомственные сети связи Украины, основой построения которых являются узлы коммутации (УК).

Экономичность коммутационной техники приобретает все большую значимость в связи с колоссальным увеличением вкладываемых в нее средств [38]. Во многом технико-экономические показатели УК определяются проектными решениями на этапах спецификации-планирования и системного проектирования. Условие снижения риска неудачных решений — обоснованный технический прогноз этих показателей. Известные способы такого прогноза [4,5,9,17,25,30,33,39,53,62] — это широко апробированные аналитические и численные методы расчета производительности УК, вероятностей блокировок, ожидания, длин очередей и т.п., а также методы машинного моделирования. Большинство из них были разработаны до массового внедрения цифровых УК и сопутствующих им терминальных устройств и потому характеризуются рядом особенностей:

1) ориентированностью на известную (фиксированную) структуру УК;
2) необходимостью совмещения специалистами знания языков и приемов программирования и знания предметной области, а также владения методами моделирования общего назначения (при машинном моделировании);
3) использованием предположений об однородности обслуживаемого трафика и стационарности его статистических характеристик;
4) ограниченным числом влияющих факторов, которые возможно учесть;
5) локальной направленностью — на анализ частей УК, а не системы в целом, часто невозможностью общего анализа УК вообще.

Решение задач синтеза и оптимизации структур УК предполагает оценку ряда альтернативных вариантов, что усложняется первой особенностью. Кроме того, далеко не для

всех типов структур УК известны способы аналитической или численной оценки. Вторая особенность затрудняет широкое инженерное применение известных методов в целях технической экспертизы и выбора конфигурации УК, ставит в зависимость качество прогноза от широты квалификации специалиста. Быстрое изменение состава оконечных устройств УК за счет роста числа персональных компьютеров и появления цифровых абонентских установок разного типа, трансформация динамики информационного обмена за счет увеличения трафика данных привело к тому, что ограничения известных методов прогнозирования часто оказываются неприемлемыми. Четвертая и пятая особенности снижают применимость известных методов к УК со сложной внутренней организацией. Во многом создание и модернизация УК производится по опыту выпуска и эксплуатации более ранних систем, часто на основе интуитивных и эвристических проектных решений. Указанные особенности снижают эффективность известных методов в изменившихся условиях эксплуатации. Возникает задача нахождения более эффективных, инженерных методов прогнозирования, применимых к цифровым УК произвольной структуры и любым условиям эксплуатации, способных учесть сложный характер взаимодействия его частей и влияние таких факторов, как сбои, отказы и т.п.

В работе предложен новый подход к проблеме прогнозирования характеристик цифровых УК, основные принципы которого таковы:

- предметная область — системы коммутации, математический аппарат — сети Петри;
- считаются известными данные функциональной спецификации УК [47,99], и решения этапа системного проектирования, определяются показатели функционирования УК;
- модель есть набор исходных данных, отражающих УК и его внешнее окружение (ВО), для использования их *универсальным* имитационным моделирующим алгоритмом;
- представление моделей графическое: сети алгоритмов, ВО, структур и определяемых характеристик УК, — с дополнением их параметрической числовой информацией;
- разделение общего графа модели на части по функциональному признаку в процессе ее построения и верификации — *функциональная декомпозиция',*
- автоматическое преобразование графической части исходных данных в цифровую информацию для моделирующего алгоритма;
- автоматизированная проверка правильности и отладка моделей.

В настоящее время нет общеизвестных универсальных методов моделирования УК *произвольной* структуры. Указанный подход позволяет отнести структуру к разряду исходных данных, которые перенастраиваются простым редактированием чертежа и файлов настройки, что позволяет строить модели произвольных структур. Универсальность моделирующего алгоритма дает возможность исключить программирование при создании моделей, что расширяет рамки использования метода, делает его доступным в инженерной практике. Иерархичность СП и их высокая выразительная мощность позволяют строить весьма сложные модели с учетом большого количества факторов влияния. Функциональная декомпозиция повышает их наглядность, облегчает верификацию и отладку за счет поэтапности, а автоматическое преобразование графической информации позволяет реализовать анализ сложную систему в целом.

Для реализации данного подхода разработаны принципы функционального и структурного описания УК с помощью нового расширения СП, а также методы анализа сетевых моделей. Предложены модификации СП:

- для отражения временных соотношений процессов в УК и ВО — синхронность и временные задержки в переходах;
- для отражения логики смены состояний и приоритетности прерываний — приоритеты;
-для отражения стохастического характера ВО — специальные позиции, устанавливающие случайную функцию возбуждения переходов.

Предложено алгоритмы работы УК отражать в модели путем трансляции SDL-диаграмм в подсеть алгоритма, ВО описывать специальной стохастической подсетью, структуры УК отображать путем выделения существенных ресурсных компонент подсистемы управления, необходимых для реализации частных алгоритмов, и создания из соответствующих им позиций подсети ресурсов. Подсети объединяются в общую сетевую модель за счет инцидентностей к переходам подсети алгоритма УК. Для учета различных факторов влияния предложено создание подсети влияний. Каждая подсеть образует слой общей модели, сопрягающийся с другими. Числовая часть модели состоит из векторов начальной разметки, приоритетов, временных задержек, распределения вероятностей случайных функций и определяется решениями этапа системного проектирования, данными спектра занятий и другой статистикой ВО. Для назначения временных задержек переходов подсети алгоритма разработаны принципы моделирования ча-

стных алгоритмов на основе принятого расширения СП. Для верификации сетевой модели проведен анализ существующих и предложены новые способы проверки ее алгоритмических свойств, сформулированы и доказаны необходимые теоремы. Нужные показатели функционирования УК предложено вычислять средствами СП в дополнительной подсети вычислений, что позволяет задавать их состав не в программе, а при построении модели. Разработана концепция динамики перемещения заявок в сети и принципы приведения технических данных к начальной разметке. Универсальный моделирующий алгоритм реализует работу сетевой модели по правилам принятого расширения СП. Для построения, анализа и испытания моделей, а также преобразования графической информации в числовую написано 8 программ. Разработанные принципы и приемы обобщены в метод, достоверность которого подтверждена сравнением результатов моделирования реальных УК со статистическими данными их работы.

ОБЩАЯ ХАРАКТЕРИСТИКА ДИССЕРТАЦИОННОЙ РАБОТЫ

Содержание работы соответствует области исследований п.2.4 Паспорта специальности 05.12.02 — “Разработка физических и математических моделей систем телекоммуникаций и управления ими, их оптимизация”.

Актуальность темы. Моделирование является необходимой частью процесса разработки и модернизации УК и позволяет значительно уменьшить затраты на оценку принятых проектных решений, а также дает возможность оценить соответствие того или иного оборудования поставленным целям в процессе коммерческой деятельности. Модернизация общегосударственных и ведомственных информационных сетей Украины связана с большим количеством как проектных, так и коммерческих решений по УК, которые составляют значительную часть стоимости этих сетей. Известные методы моделирования УК и им подобных систем ориентированы на анализ устройств с фиксированной архитектурой, что затрудняет их использование для сопоставления многих возможных вариантов. Традиционные модели, как правило, рассматривают составляющие УК отдельно, в результате чего теряется влияние фактора их взаимосвязи. Для большинства из них необходимо соблюдение ряда упрощающих предположений, которые становятся неприемлемыми в связи с изменением состава коммутируемого трафика. Ряд методов, например широко известные системы массового обслуживания, требует после-

довательного характера функционирования объекта, и не пригоден для описания параллельных потоков задач УК. Значение имеет и тот факт, что большинство известных методов моделирования ненаглядны и требуют достаточно высокой квалификации от специалиста не только в области коммутационной техники, но и в программировании. Все это снижает качество принимаемых проектных и коммерческих решений при организации современных сетей связи и вызывает необходимость поиска новых методов моделирования УК.

Связь работы с научными программами, планами, темами. Настоящая работа проводились в соответствии с Концепцией построения цифровой сети связи железнодорожного транспорта [32], ее результаты могут быть использованы для технической экспертизы проектных решений при модернизации существующей сети связи в рамках реализации Генеральной схемы цифровой интегральной сети связи железнодорожного транспорта Украины [32].

Цель и задачи исследования. Цель проведенной работы — разработка принципов и общего метода моделирования проектируемых и существующих цифровых узлов коммутации произвольной структуры для прогноза их основных качественных показателей при функционировании в условиях современных сетей связи.

Основная идея работы — использование функционального и структурного описания УК средствами специального расширения сетей Петри и автоматическое преобразование графических образов СП в исходные данные для универсального имитационного моделирующего алгоритма.

Для достижения поставленной цели и реализации идеи определены такие задачи:

- разработать принципы функционального описания алгоритмов работы УК, позволяющие транслировать SDL-диаграммы алгоритмического обеспечения в графы СП;
- разработать принципы функционального описания в терминах СП схем случайных событий во внешнем окружении УК, адекватно его алгоритму работы;
- разработать принципы структурного и функционального описания в терминах СП внутренней организации конкретных реализаций АО и ПО УК;
- отыскать способы качественного анализа сетевых моделей УК для их отладки, верификации, проверки конструктивности реализуемых УК алгоритмов;

- отыскать способы количественного анализа сетевых моделей УК для получения обоснованного прогноза на интересующие исследователя показатели работы УК;
- обобщить разработанные принципы описания и способы анализа, с учетом влияния дополнительных факторов, в виде метода моделирования цифровых УК;
- автоматизировать получение матриц инцидентности по графу сетевой модели, реализовать программно универсальный имитационный моделирующий алгоритм, провести испытания на конкретных образцах УК и внедрение полученных результатов.

Методологический аппарат исследований: теория сетей Петри, теория моделирования сложных систем, теория телетрафика, теория графов, языки программирования, теоретические приложения цифровой коммутации.

Научная новизна полученных результатов. Предложено и исследовано новое расширение СП, удобное для создания сетевых моделей УК, которое, в отличие от известных, включает случайные управляющие приоритетные функции с произвольным законом распределения. Впервые сформулированы и доказаны основные теоремы, необходимые для анализа таких сетей.

Впервые предложен и программно реализован способ анализа указанного расширения СП на основе построения покрывающих деревьев сети с матричным, а не векторным представлением вершин.

Получил дальнейшее развитие принцип трансляции SDL-диаграмм алгоритмов функционирования УК в графы СП, обеспечивающий теперь получение удобного для анализа, структурно ограниченного подкласса автоматных СП.

Получил дальнейшее развитие принцип моделирования случайных потоков вызовов УК, который теперь позволяет учесть взаимосвязь моментов поступления заявок на обслуживание разного типа с самим процессом их обслуживания УК.

Впервые предложен и программно реализован принцип многослойности сетевой модели УК, основанный на функциональной декомпозиции СП, который сделал возможным более подробное его описание при сохранении наглядности графических образов и возможности автоматического построения матриц инцидентности.

Разработан новый метод построения и испытания многослойных сетевых имитационных моделей УК с использованием предложенного расширения СП и современных графических сред со встроенными языками программирования.

В целом, научное значение работы состоит в развитии принципов сетевого моделирования на основе СП для объектов со сложной аппаратно-программной структурой и стохастическим внешним окружением, с учетом особенностей цифровых УК.

Обоснованность и достоверность полученных результатов подтверждается: испытанием моделей реальных УК, сравнением полученного прогноза с техническими данными и данными статистики УК, находящихся в эксплуатации; успешными результатами внедрения на заинтересованных предприятиях; апробацией на научных конференциях и семинарах; публикациями в научных журналах и сборниках.

Практическое значение полученных результатов. Разработанный метод моделирования может быть применен как:

- основа технической экспертизы проектных и коммерческих решений при выборе оборудования в технико-экономических обоснованиях и рабочих проектах по модернизации существующих и созданию новых сетей связи;
- средство оценки проектных решений при разработке новых и модернизации существующих АО и ПО цифровых УК;
- элемент системы автоматизированного проектирования АО и ПО цифровых УК.

На основе полученных в работе результатов написан и отлажен пакет программ, который совместно с графической средой AutoCAD R14 (R15) [80] предназначен для построения, анализа, отладки и испытания имитационных моделей цифровых УК и может быть использован заинтересованными организациями.

Разработанный метод моделирования и указанный пакет программ были успешно применены для технической экспертизы цифровых УК на предприятиях: ОАО производственно-технологической связи Министерства угольной промышленности Украины “Укруглетелеком” (акт о внедрении “Укруглетелекома” от 29.01.99); ЗАО “Фарлеп-Телеком-Холдинг” (акт о внедрении “Фарлеп-Телеком-Холдинга” от 27.01.99). Результаты моделирования конкретных УК позволили принять обоснованные решения по выбору оборудования для приобретения и его оптимальной конфигурации в конкретных условиях. Метод и пакет программ получили положительные отзывы экспертных комиссий на указанных предприятиях и рекомендованы к использованию.

Личный вклад соискателя. Автором лично предложено указанное расширение сетей Петри для функционального и структурного описания УК и обоснованы принципы такого описания; разработаны принципы описания случайных потоков заявок посредством СП, трансляции SDL-диаграмм в СП, многослойности сетевой модели; разработаны и программно реализованы принципы автоматического построения матриц инцидентности и испытания общей сетевой модели.

Апробация результатов диссертации. Результаты диссертационной работы доложены, обсуждены и получили одобрение на кафедре “Транспортная связь” Харьковской государственной академии железнодорожного транспорта (ХарГАЖТ), на 60-й научно-технической конференции ХарГАЖТ (с международным участием, 1998г.), на XI научной конференции “Перспективные системы управления на железнодорожном транспорте” (г.Алушта, 1998г.), в докладах на межкафедральных семинарах ХарГАЖТ и Харьковского государственного технического университета радиоэлектроники.

Публикации. По теме диссертации опубликовано пять научных статей: две в научно-техническом журнале и три в сборниках научных трудов.

РАЗДЕЛ 1

ОБОСНОВАНИЕ РАЗРАБОТКИ МЕТОДА МОДЕЛИРОВАНИЯ ЦИФРОВЫХ УЗЛОВ КОММУТАЦИИ

1.1 Задачи метода моделирования узлов коммутации и требования к нему

Стремительное развитие средств коммутационной техники, выполнение ими новых, не свойственных ранее функций, высокая конкуренция на рынке их сбыта, изменения в структуре трафика и его рост ужесточают требования к разрабатываемым вновь и модернизируемым узлам коммутации (УК). Современные системы коммутации в основном цифровые, различия заключаются в назначении, месте в системе распределенной коммутации: абонентские концентраторы, контейнерные абонентские, учережденческо-производственные, опорные коммутационные станции, узлы коммутации интегральных сетей и др., а так же во внутренней организации самих средств. Последняя, наряду с компонентной базой, определяет технико-экономические показатели. Характерной особенностью современных узлов коммутации является высокая серийноспособность, то есть возможность экономически выгодного наращивания емкости в широких пределах, причем аппаратное (АО) и программное (ПО) обеспечение, как правило, может конфигурироваться под конкретные потребности заказчика. Вопреки некоторым прогнозам [69] номенклатура коммутационной техники остается достаточно разнообразной как по техническим решениям и реализуемым алгоритмам, так и по стоимости. Такое положение вещей ставит следующие основные задачи на рынке систем коммутации:

- перед потребителями — взвешенной, аргументированной оценки того или иного коммутационного оборудования с точки зрения его соответствия имеющимся потребностям и возможностям;

- перед продавцами — обоснованного выбора нужной конфигурации узла коммутации и получения доверительного прогноза на основные его качественные показатели при функционировании в конкретных специфических условиях заказчика;

- перед разработчиками — нахождения новых, наиболее экономичных структур АО и ПО, которые характеризовались бы оптимальной избыточностью, высокой унификацией и т.д., а также уточнение влияния того или иного новшества на качественные показатели УК при модификациях (модернизациях) его аппаратного и/или программно-

го обеспечения. Решение всех трех указанных задач немыслимо без объективной предварительной оценки принятых решений, сравнения возможных вариантов между собой.

Исследование работы некоторой системы можно в принципе осуществить либо на опытном образце, либо испытанием (анализом) ее модели. УК по сложности и стоимости можно отнести к классу достаточно больших систем, для которых, создание или приобретение опытных образцов требует немалых затрат. Моделирование позволяет значительно уменьшить затраты на оценку проектных или коммерческих решений, и является необходимой частью процесса разработки и модернизации подобных систем. В соответствии с принятым на сегодня системным подходом к созданию новых сложных устройств, процесс проектирования УК можно разбить на два крупных этапа:

- системного проектирования (макропроектирования, внешнего проектирования), результатом которого должно быть определение основной структуры УК, состав модулей и межмодульных интерфейсов;
- технического проектирования (микропроектирования, внутреннего проектирования) то есть непосредственной разработки УК по полученной структуре и его отладки.

Этап системного проектирования предполагает [53,62,68] определение исходных данных, построение модели внешней среды, возможных моделей самого УК и на основании этих моделей — исследование общей математической модели, в результате чего и определяется его основная структура, как наиболее оптимальная. Задача оптимизации в приложении к системам, подобным УК, может быть решена только путем сопоставления различных вариантов их внутренней организации при заданных характеристиках потоков решаемых задач и приоритетности требований к системе. Для этого необходимо разработать модели и методы оценки различных вариантов их реализации с точки зрения заданной критериальной базы. Кроме того, обычно число всевозможных таких вариантов велико, и полный перебор их, как правило, нельзя осуществить из-за ограниченности вычислительных ресурсов. Таким образом возникают задачи: 1) адекватного формализованного описания оцениваемого объекта, которое обеспечивало бы, с одной стороны — возможность изменения его структурных и функциональных параметров (перенастройку) для перехода от одного альтернативного варианта к другому, а с другой стороны — возможность оценки варианта по критерию; 2) ограничения множества рассматриваемых вариантов практической реализации системы с тем, чтобы обеспечи-

валась физическая возможность их анализа, и не был потерян оптимальный вариант. Как видно, первая задача практически означает построение гибкой модели системы. Моделирование успешно применяется и для решения второй задачи [62,67,68].

Традиционный подход к построению моделей УК предусматривает описание алгоритмов функционирования при помощи SDL-диаграмм [47,99,100], на основе которых разрабатывается ПО УК [7]. Функциональное описание коммутационных полей и определение потребного числа обслуживающих пассивных приборов производится методами теории телетрафика, которые в основном предложены в эпоху электромеханических АТС (графы Ли, методы Якобеуса, Клоза, эффективной доступности и др. [30,33,75]). При этом предполагается, что архитектура подсистемы управления заранее известна, и необходимо определить количества ее составляющих или производительность всего УУ. Такой подход основан на особенностях электромеханических УК и централизованных электронных с пассивными периферийными устройствами, и он неудобен при рассмотрении УК других типов. В [30] рассмотрены аналитические модели коммутационных систем и управляющих устройств (УУ). Такие модели предполагают серьезные упрощающие допущения — пуассоновский характер потоков заявок, постоянство длительностей набора номера, разговора и т.п., что в условиях изменившейся структуры трафика не всегда приемлемо. Главный недостаток таких моделей — узкая направленность на УУ одного типа или структуры, невозможность учета вариантов разделения функций между его составляющими, сложного характера их их взаимодействия. Так например расчет производительности УУ УК в [30] предлагается вести при условиях: исполняется только установление соединения по системе с ожиданием, длительность попыток постоянна, длина очереди неограниченна, поток вызовов простейший. Получение аналитической зависимости основано на однофазной модели системы массового обслуживания (СМО) [4,19,24,62] вида M/M/1/oo. Использование двухфазных СМО [62] требует еще больших, дополнительных упрощений: экспоненциальные обслуживающие приборы с постоянной интенсивностью, последовательный характер их работы, потоки занятий и освобождений простейшие, интенсивности их равны, приборы укрупненяются объединением и т.п. [62], — в противном случае получение характеристик невозможно или неоправданно трудоемко. СМО мало пригодны для устройств, работающих в реальном масштабе времени и с параллельными непуассоновскими потоками задач, каковыми

являются современные УК. Кроме того существуют различные методики моделирования микропроцессорных систем общего назначения [11,39,40,44,61,62,86], которые не учитывают специфики УК, и потому трудно применимы к ним. В основном они предполагают написание специальной моделирующей программы для всякой новой структуры, что затрудняет их широкое инженерное использование. Общими недостатками рассмотренных вариантов моделей являются: раздельное рассмотрение структуры, ресурсов и алгоритма работы системы; ориентированность на фиксированную структуру, что приводит к необходимости составления всякий раз новой моделирующей программы для оценки очередной рассматриваемой структуры УК.

Для преодоления указанных недостатков необходим такой метод моделирования УК, который предусматривает возможность формализованного задания *всех* имеющихся на этапе системного проектирования исходных данных и обеспечивает получение интересующих показателей функционирования в удобном виде и с приемлемой точностью. Тогда требования к методу моделирования УК сформулируем в виде задачи: разработать принципы построения моделей УК, которые дали бы возможность:

1) оценить эффективность и основные показатели качества функционирования УК,
2) учесть одновременное действие на пропускную способность системы:
   - структуры подсистемы управления, величины ее ресурсов, алгоритма работы,
   - сложных вероятностных характеристик потоков входных заявок из внешнего окружения, взаимосвязь различных заявок между собой,
   - блокировок в коммутационном поле,
   - наличия нескольких параллельных потоков задач;
3) легко перенастраивать модели на различные:
   - условия функционирования во внешнем окружении,
   - конфигурации подсистемы управления,
   - алгоритмы функционирования,
   - определяемые показатели качества обслуживания;
4) отнести структуру системы к числу исходных данных с формализованным заданием, исключить процесс написания программ из моделирования;
5) относительно просто строить и эксплуатировать модель, получать результаты в удобном виде.

1.2 Выбор средств моделирования

Под моделью понимается физический объект или некоторое описание отображающее те свойства и отношения оригинала (моделируемого объекта), которые в данном исследовании признаны существенными. Очевидно моделирование является в большинстве случаев более приемлемым вследствие трудностей технического и экономического характера, с которыми связаны натурные испытания.

Среди возможных моделей можно выделить объектные (физические объекты-макеты и др.), словесно-опсисательные и математические. Ясно, что модели первого рода, в применении к УК, практически столь трудно реализуемы, что их использование не имеет смысла. Словесно-описательные модели (техзадания) представляют собой совокупность эмпирических понятий — неформализованный текст, рисунки, характеризующие оригинал в самых общих чертах. Такие модели так же невозможно непосредственно использовать для исследований и формальных методов анализа.

Практическое использование для моделирования УК имеют математические модели описывающие оригинал переменными или другими условными знаками и реализуемые на ЭВМ. Будем различать:

1) аналитические модели, в которых интересующие характеристики можно получать либо в явном виде, либо числено, либо качественно (оценка свойств решения);

2) имитационные модели в которых воспроизводится функционирование оригинала в формальном виде, причем значения описательных переменных равны (пропорциональны) соответствующим параметрам оригинала.

Анализ особенностей этих двух основных типов математических моделей позволяет сделать вывод о том, что если в качестве моделируемого объекта выступает УК, то аналитические модели, которые удовлетворительно описывали бы УК в целом, практически неосуществимы даже при значительных упрощающих предположениях. Аналитические методы нашли широкое применение для моделирования отдельных частей УК: ступеней искания коммутационной системы, отдельных управляющих устройств и т.д. [30,33]. Однако подход к оценке характеристик УК, при котором оцениваются его части, а затем делается вывод о характеристиках УК в целом имеет тот очевидный недостаток, что теряется влияние взаимосвязи частей на свойства системы в целом. Наиболее уни-

версальным и пригодным для анализа существующих УК является метод имитационного (статистического) моделирования, поскольку он позволяет решить задачу определения качественных показателей УК практически любой сложности и с приемлемой точностью при известных параметрах потока заявок, структурной схеме и других данных об УК. Имитационная модель позволяет учесть влияние большого количества внешних и внутренних факторов, сложный характер взаимодействия отдельных частей системы. Несмотря на значительно большее требуемое время моделирования, имитационные модели практически вытеснили модели других типов при исследовании систем подобных УК [9,11,48-53,60-62,67,68,74].

Для построения имитационной модели прежде всего необходимо представить в формальном виде моделирующий алгоритм, описывающий функционирование УК как дискретной динамической системы. Важное место в теории дискретных устройств занимает понятие автомата и основывающиеся на этом понятии методы описания дискретных систем объединенные в рамках теории конечных автоматов: таблицы переходов и выходов, графы состояний автомата, матричные и логические схемы алгоритмов и т.д. [39,40]. Эти методы формального описания дискретных систем предполагают: 1) последовательный способ функционирования системы — переход из состояния в состояние; 2) достаточно малое общее количество состояний, при котором еще применимы методы анализа указанной теории. Совершенно очевидно, что УК, как дискретные системы, не отвечают таким требованиям, поскольку в них можно выделить несколько независимых параллельных потоков задач, которые иногда взаимодействуют: потоки исходящих, внутренних, входящих вызовов, потоки заявок на дополнительные виды обслуживания и т.д., то есть функционирование УК представляет собой ряд параллельных независимых взаимодействующих процессов. Обслуживание вызовов одного из параллельных потоков в многопрограммном режиме с разделением времени представляет собой параллельные процессы конвейерного типа [34]. Кроме того, общее количество возможных состояний УК, даже небольшой емкости, измеряется тысячами, что делает невозможным его описание методами теории конечных автоматов.

Другими распространенными формализмами, используемыми для описания дискретных динамических систем являются агрегативные системы [9,39], счетчиковые автоматы и машины Минского [46], машины Тьюринга [35], различные схемы параллель-

ных программ (Карпа-Миллера, А-программы Котова-Нариньяни и т.п.), а так же сети Петри.

Агрегативные модели предполагают представление отдельных элементов системы агрегатами, каждый из которых характеризуется набором переменных и операторов их преобразования. Для адекватного представления таким образом УК потребуется большое число агрегатов, сложные операторы преобразования, что делает процесс создания моделей реального времени с параллельными потоками задач трудоемким, а в ряде случаев нереализуемым.

Абстрактные машины Минского и Тьюринга — математические формализмы используемые главным образом в теории схем программ [35], теории параллельных вычислений. Вместе с упомянутыми схемами параллельных программ, абстрактные машины имеют достаточно узкую “вычислительную” направленность, перенесение свойств машин на программы позволяет получить решение разнообразных задач программирования. Иллюстративные возможности этих формализмов невелики, однако моделирующие свойства абстрактных машин — способность к адекватному отображению существенных свойств и внутренних взаимосвязей сложной дискретной динамической системы — наивысшие среди перечисленных моделей [34]. По отношению к УК абстрактные машины более подходят к описанию алгоритмической части, но совершенно неудобны для отображения сложных взаимосвязей частей УК, как системы. Подобные замечания действительны и для других упомянутых схем параллельных программ.

Наиболее удобным математическим формализмом для описания УК представляются сети Петри (СП) потому что:

1) СП легко моделируют асинхронные параллельные взаимодействующие процессы, отражая общую динамику работы дискретной динамической системы.
2) СП удобно отражать возможности многих других механизмов, предложенных для описания параллельных систем (семафоры Дейкстры, ресурсы и др.).
3) СП допускают любую смысловую интерпретацию своих составляющих, что позволяет моделировать например потоки заявок, аппаратные средства, причем совместно, в рамках одной модели. Это свойство обуславливает широкую применимость СП вообще и удобство для описания УК в частности.

4) СП допускают различную трактовку своих элементов по уровню абстракции (детализации), что позволяет организовать ее иерархическое построение рассматривая, например, некоторый переход как подсеть более низкого уровня иерархии. Это свойство значительно сглаживает противоречие между требованиями простоты и адекватности модели.
5) СП по моделирующим возможностям значительно превосходят конечные автоматы и приближаются к абстрактным машинам, уступая последним. В [34] моделирующие возможности формализованы в понятии “выразительная мощность” полученное на основе рассмотрения формальных языков, порождаемых моделью. Здесь же доказано, что обычные сети Петри занимают промежуточное положение по выразительной мощности между конечными автоматами и машинами Тьюринга, а некоторые классы (расширения) СП (ингибиторные, приоритетные) равномощны последним.
6) Графы СП обладают высокой иллюстративностью и дают наглядное представление о протекании процесса. В [11] показано, что граф СП достаточно сложной системы значительно меньше диаграммы переходов состояний автомата.
7) Применимость СП сравнительно легко расширяется путем их модификации, позволяющей учесть влияние тех или иных существенных особенностей оригинала (время, приоритеты, параметры меток и др.). В результате появилось большое количество модификаций СП предназначенных для решения того или иного круга задач: временные, сети Мерлина, приоритетные, ингибиторные, самомодифицирующиеся, синхронные, раскрашенные, числовые (оценочные, или Е-сети), О-сети (сети-процессы), нагруженные, управляющие, аппаратные, регулярные, иерархические, и многие другие, а так же их объединения.

Основной целью представления УК в виде СП представляется получение удобного и наглядного моделирующего алгоритма, на основе которого может быть составлена *универсальная* имитационная программа. Это позволяет исключить из моделирования процесс программирования путем замены его наглядным графовым описанием УК (чертежом) и использования универсальных моделирующих программ, что упрощает инженерные приложения таких моделей.

В случае разработки нового УК, когда моделируемый объект еще не существует, алгоритм его работы не отлажен — желательной является предварительная проверка

моделирующего алгоритма на отсутствие критических свойств — тупиков, зацикливаний и т.п. как с точки зрения правильности алгоритма работы проектируемого УК, так и корректности самой модели. Известно, что в теории СП разработаны достаточно эффективные методы анализа сетей на основные алгоритмические свойства: живости, ограниченности (безопасности), достижимости некоторой разметки и др., которые могут быть соответственно интерпретированы. Таким образом аппарат СП позволяет специальными методами проверить конструктивность построенной модели еще до начала ее статистических испытаний.

Моделирование в процессе проектирования есть не только способ анализа, но и средство отладки АО и ПО УК, определения “узких мест”, ошибок и неудачных проектных решений. Особое значение в этом смысле приобретает иллюстративность моделей на основе СП. Простота понимания и быстрота восприятия графических образов СП создает предпосылки активного использования машинной графики в моделировании. Наглядность путей перемещения меток в сочетании с возможностями цветного графического дисплея позволяет непосредственно наблюдать за ходом процесса и производить качественную оценку как модельных, так и проектных решений, что в целом повышает эффективность моделирования. Использование современных графических сред значительно облегчает процесс построения самой модели на основе СП, ее отладку, проверку на корректность, настройку на различные условия функционирования.

На этапе технического проектирования прежде всего встает задача разработки функциональных схем каждого блока УК (логическое проектирование). При использовании СП в ряде случаев можно получить граф сравнительно небольшой размерности, что позволяет применить к такой СП аналитические методы определения основных ее характеристик [34,62,68] и тем самым уменьшить объемы имитационного моделирования и сократить время разработки блока.

Описанные возможности СП, очевидно, позволяют использовать их и для решения первой из указанных выше (см. п.1.1) задач оптимизации [11,40,62,66,67,68]. Распространенной и небезосновательной является сложившаяся точка зрения о том, что наилучших результатов в решении задачи ограничения множества рассматриваемых вариантов можно достичь сочетанием различных подходов:

- типизация структур, учет опыта проектирования подобных систем ранее;

- умозрительные (а иногда даже интуитивные) соображения по определению явно неоптимальных вариантов, или вариантов с незначительными отличиями на основании стоимостных, массогабаритных, надежностных, климатических ограничений, доступности для заказчика, перспективности и т.п.;
- приблизительная, упрощенная предварительная оценка вариантов, не требующая значительных затрат времени и вычислительных ресурсов.

Как видно СП могут быть использованы в последнем случае, то есть возможно их применение и для решения второй из указанных задач оптимизации.

По сравнению с большинством других методов формального описания сложных систем, СП обладают по-видимому наибольшей универсальностью, являясь, по сути, абстрактной условно-событийной моделью, применимой практически к любой системе [59]. На основании приведенных рассуждений принято решение о целесообразности использования математического аппарата СП для построения имитационных моделей аппаратно-программной структуры и внешнего окружения цифровых УК. Данный выбор позволяет конкретизировать принципы метода моделирования:

1) предметная область — системы коммутации, математический аппарат — сети Петри;

2) считаются известными (заданными) данные функциональной спецификации УК и решения этапа системного проектирования: структурная схема, алгоритм работы в терминах SDL, данные о производительности структурных единиц, строение и режим использования памяти, техническое описание процесса установления соединений, статистика внешнего окружения УК;

3) определяются задаваемые показатели функционирования УК, конструктивность реализуемого им алгоритма;

4) модель есть набор исходных данных, отражающих УК и его внешнее окружение (ВО), для использования их *универсальным* имитационным моделирующим алгоритмом;

5) содержание модели — граф СП и числовая информация о динамике ее работы: начальная разметка, временные параметры и др.;

6) сложность оригинала модели порождает большой объем графа СП, что требует автоматического преобразования графической части исходных данных в цифровую информацию для моделирующего алгоритма;

7) использование методов анализа СП с целью автоматизации проверки правильности и отладки моделей.

ВЫВОДЫ

С целью обоснования разработки метода моделирования цифровых УК, в разделе рассмотрены основные задачи стоящие перед участниками рынка коммутационного оборудования. Выделено три класса таких задач:

- перед потребителями — взвешенной, аргументированной оценки коммутационного оборудования с точки зрения его соответствия имеющимся потребностям и возможностям;

- перед продавцами — обоснованного выбора нужной конфигурации узла коммутации и получения доверительного прогноза на основные его качественные показатели при функционировании в конкретных специфических условиях заказчика;

- перед разработчиками — нахождения новых, наиболее экономичных структур АО и ПО УК и уточнение влияния того или иного новшества на качественные показатели УК при модернизациях его аппаратного и/или программного обеспечения.

Для решения указанных задач необходима объективная предварительная оценка проектных и коммерческих решений, сравнение возможных вариантов реализации УК, что практически возможно реализовать только путем моделирования.

Анализ известных методов моделирования показал, что традиционные методы, разработанные ранее, не позволяют в полной мере решать эти задачи для современных УК, потому что:

1) создание приемлемых аналитических моделей для цифровых УК со сложной аппаратно-программной структурой практически не возможно, так как требует недопустимо больших упрощений при формальном описании как самих УК, так и их внешнего окружения, особенно в условиях изменившегося состава информационного трафика;
2) традиционные методы расчета пропускной способности систем коммутации ориентированы на фиксированную структуру УК, что затрудняет решение задачи ее оптимизации;
3) методы моделирования микропроцессорных систем общего назначения не учитывают особенностей коммутации, предполагают создание моделирующих

программ для всякой новой структуры УК, что затрудняет широкое инженерное использование специалистами коммутационной техники.

Поэтому поставлена задача разработки нового метода, к которому предъявлены требования: учета одновременного действия на пропускную способность системы разнообразных факторов, характерных для современных цифровых УК; легкой перенастройки модели; исключения процесс написания программ из моделирования; простоты построения и эксплуатации модели, удобства получаемых результатов.

Для выбора средств моделирования проанализированы альтернативные традиционным СМО, формальные математические схемы, которые могут быть использованы в таких целях. Установлено, что наилучшим образом поставленной задаче отвечают сети Петри, так как они легко моделируют асинхронные параллельные, отражать возможности многих других схем для описания параллельных систем, допускают любую смысловую интерпретацию своих составляющих и любой уровень детализации, благодаря иерархичности, имеют высокую выразительную мощность, обладают высокой иллюстративностью и наглядностью, легко расширяется, что позволяет учесть влияние тех или иных существенных особенностей оригинала (время, приоритеты, и др.), позволяют формализовать разнородные исходные данные в рамках одной модели.

Поскольку поставлена задача общего анализа УК в целом, предложено в качестве исходных данных принять данные функциональной спецификации и решения этапа системного проектирования, которые достаточно полно характеризуют УК. Таким образом общая концепция метода предполагает: специализацию по системам коммутации; математический аппарат — СП; исходные данные — функциональная спецификация, решения этапа системного проектирования, статистика ВО; задание состава показателей функционирования УК, проверку конструктивности реализуемого им алгоритма; модель есть набор исходных данных для использования их *универсальным* имитационным моделирующим алгоритмом; содержание модели — граф СП и числовая информация о динамике ее работы: начальная разметка, временные параметры и др.; необходимость автоматического преобразования графической части исходных данных в цифровую информацию для моделирующего алгоритма; использование методов анализа СП с целью автоматизации проверки правильности и отладки моделей.

РАЗДЕЛ 2

РАЗРАБОТКА МЕТОДИКИ ПОСТРОЕНИЯ СЕТЕВЫХ МОДЕЛЕЙ ЦИФРОВЫХ УЗЛОВ КОММУТАЦИИ

2.1. Сетевое описание алгоритмов функционирования

2.1.1. Общие алгоритмы функционирования узлов коммутации

Как было указано выше, наиболее эффективным средством функционального описания УК является аппарат сетей Петри, сочетающий способность к описанию параллельных независимых конкурирующих процессов и наглядность модели. Рассмотрим задачу описания в терминах СП общих алгоритмов функционирования УК.

Исходным пунктом для модели является состав реализуемых процессов, входных и выходных сигналов, состояний и алгоритмы их смены, который в соответствии с рекомендациями МККТТ на этапах спецификации-планирования и системного проектирования должен быть представлен в терминах SDL [47]. Пример упрощенной SDL-диаграммы, описывающей процесс обслуживания местного вызова, приведен на рис.2.1. Эта диаграмма построена на базе автоматной модели УК, реализующего процесс обслуживания одиночного вызова [77,99,100].

Непосредственное использование языка SDL для создания имитационных программ с целью оценки качественных показателей той или иной возможной реализации узлов коммутации (УК) затруднено, поскольку во-первых, средствами SDL нельзя отобразить обслуживание потока заявок, а не одиночного вызова; во-вторых SDL описывает алгоритм обслуживания вызова абстрактным УК любой возможной реализации и не имеет средств задания отличительных особенностей конкретного УК. Это следует из того, что SDL рассматривает УК как "черный ящик" — автомат, определенным образом реагирующий на входные сигналы. Такое представление удобно для разработки алгоритмического обеспечения УК [7], но не для имитационного моделирования с описанием параллельных потоков задач и структур конкретных УК. Сеть Петри лишена указанных недостатков и является общепризнанным средством имитационного моделирования [59,62,66]. В [59] показано, что блок-схема алгоритма программы эквивалентна графу СП, если вершины блок-схемы заменить дугами сети Петри, а дуги блок-схемы — вершинами СП при соблюдении некоторых правил. Однако SDL-диаграмма не является

блок-схемой алгоритма программы, и хотя она и задает алгоритм обработки вызова, такое преобразование не применимо для перехода к СП в силу следующих обстоятельств.

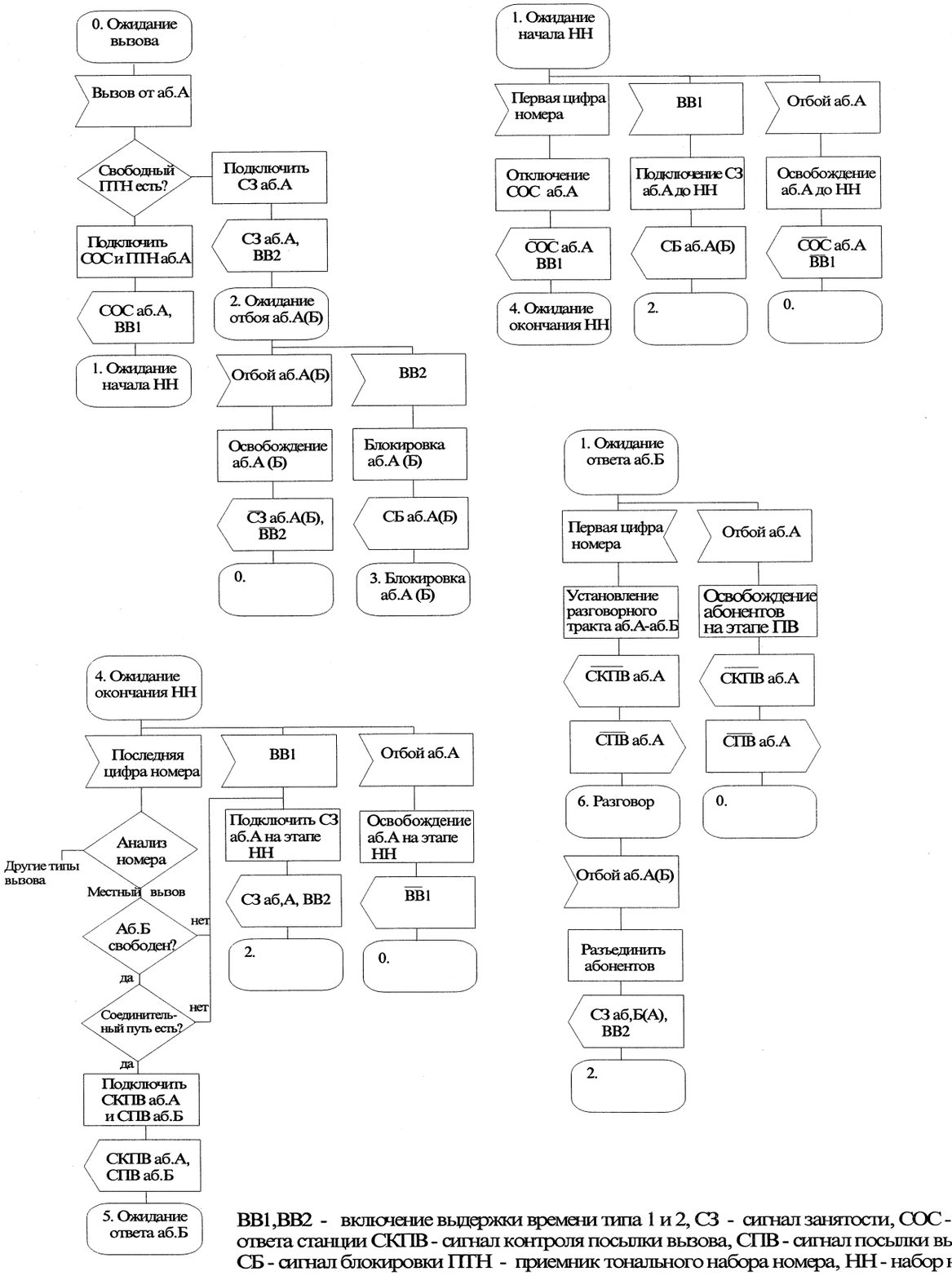
1. Позиции SDL-диаграммы имеют различный функциональный смысл и не могут быть однозначно отображены дугами сети (например ЗАДАЧА и РЕШЕНИЕ).

2. Топология SDL и сети Петри различна. В SDL диаграмме обязательно указываются вершины с выходными сигналами во внешнюю среду (ОС, КПВ, СЗ, СУВ...), которые располагаются последовательно с вершинами состояний УК, как автомата. Эти сигналы могут инициировать какие-то процессы во внешней среде, но не будут участвовать в дальнейших фазах обслуживания вызова. Поэтому в месте их возникновения должно быть разветвление процесса, описываемого сетью, а не последовательное, как в SDL, расположение вершин (см. пример на рис.2.2).

Для перехода от SDL- диаграммы к сети Петри можно использовать соответствия между понятиями теории конечных автоматов, на которой базируется SDL, и понятиями теории сетей Петри. Анализ основных определений языка SDL позволяет провести следующие соответствия между ними [7,39,77]:

- ВХОД — множество входных сигналов характеризующих окружение УК (участников соединений);
- ВЫХОД — множество выходных сигналов, вырабатываемых для окружения;
- СОСТОЯНИЕ — множество устойчивых внутренних состояний УК ;
- РЕШЕНИЕ — множество неустойчивых внутренних состояний из которых возможен переход в одно из нескольких альтернативных состояний в зависимости от значений управляющего сигнала;
- ЗАДАЧА — функции переходов и выходов. Содержание задач определяет порядок взаимодействия УК с внешним окружением и функционирования структурных единиц УК (коммутационной системы, станционных комплектов, ПНИ и т.д.).
- ЗАПОМИНАНИЕ — с точки зрения рассматриваемых аналогий, элемент эквивалентен ЗАДАЧЕ специфического содержания.

Определения СИГНАЛ и ПЕРЕХОД являются объединением перечисленных выше определений графические символы им не присвоены и для сетевого описания использовать их не будем.



ВВ1,ВВ2 - включение выдержки времени типа 1 и 2, **СЗ** - сигнал занятости, **СОС** - сигнал ответа станции **СКПВ** - сигнал контроля посылки вызова, **СПВ** - сигнал посылки вызова, **СБ** - сигнал блокировки **ПТН** - приемник тонального набора номера, **НН** - набор номера.

ВВ,СЗ,СОС,СКПВ,СПВ - отключение соответствующих сигналов.

Рис.2.1. Пример упрощенной SDL-диаграммы

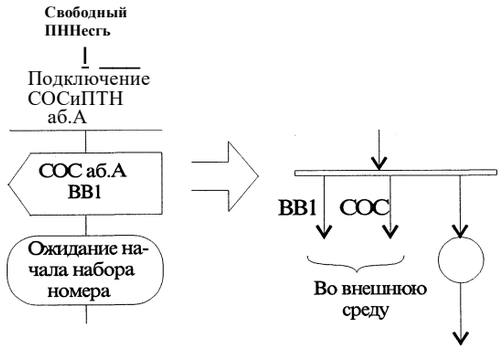


Рис.2.2. Пример разветвления сети

Дадим описание процесса трансляции SDL-диаграммы в сеть Петри на языке множеств.

Формальное описание SDL диаграммы есть множество *SDL:*

$$SDL = \{I,O,S,R,P,fg\},$$

где *O* — конечное множество символов ВЫХОД,

S — конечное множество символов СОСТОЯНИЕ,

R — конечное множество символов РЕШЕНИЕ,

P — конечное множество символов ЗАДАЧА:

$$I^* 0;\ O * 0;\ S * 0;\ R^*0;\ P * 0;$$

$$InOnSnRnP = 0;.$$

f — функция следования

$$f: (S\ xI \text{-}> \{0;1\})\ л((1u0\ uP)\ x\ (O\ uS\ uRuP)$$

$$A(RX(O\ uSuRuPj \text{ -}> \{0;1\})$$

В выражении отражены правила следования символов в SDL диаграммах [7,47,77]

g — функция предшествования

g.- *(IxS~>{0;1})*

Поскольку язык SDL построен на базе автоматной модели УК, реализующего процесс обслуживания одиночного вызова, SDL-диаграмму можно использовать для задания логической последовательности событий, которые должны быть реализованы алгоритмом функционирования УК (обслуживание вызовов, сервисные услуги, тестирование и т.д.). Размеченная сеть Петри, отражающая такую последовательность в силу своих динамических свойств позволяет проследить потоки заявок, а не обслуживание одиночного вызова. При этом состояние УК в автоматной модели будет соответствовать фазам (этапам) обслуживания заявок. Под заявкой на обслуживание понимается

любое изменение состояния точек сканирования несущее информацию о входных сигналах из внешнего окружения или отсутствие изменения в течение заданного времени.

Как известно обычную размеченную сеть Петри формально можно задать как множество:

$$N = \{ВД), Ф, Н, Мо\},$$

где B — конечное множество позиций, $B * 0$,

$$B\text{-}\{Ь_ь\ Ь_2,\ Ь_3,\ldots,\ Ь_{,,}\};$$

D — конечное множество переходов, $D * 0$

$$D=\{d_{It}\ d_2,\ d_3,\ldots,\ d_m\};$$

где n, m — натуральные.

$Ф$ — прямая функция инцидентности:

$$Ф: B\ xD\ \{0;1\}.$$

H — обратная функция инцидентности:

$$H:DxB\text{-}>\ \{0,1\}.$$

$Мо$ — начальная разметка сети.

Логический смысл позиций сети — условия наступления событий, отображаемых переходами. Если в качестве события принять фазу обслуживания заявки, то входные сигналы и нахождение заявки на (н-1)-й фазе обслуживания будут условиями для события "переход заявки к и-й фазе обслуживания".

Пусть каждой позиции некоторой сети соответствует определенная фаза обслуживания заявок. Выше замечено, что любое состояние УК, как конечного автомата соответствует фазам обслуживания потока заявок. Отсюда следует, что для перехода к описанию алгоритма функционирования УК сетью Петри можно каждому символу SDL описывающему состояние УК (устойчивое или неустойчивое) поставить в соответствие позицию сети, то есть:

$$(SuR) ->B.$$

При этом необходимо учитывать, что в SDL-диаграммах часто одна и та же вершина дублируется в разных местах во избежание затемнения чертежа (например "ожидание вызова" на рис.2.1). Различные копии этой вершины говорят не о дополнительных вершинах SDL, а о дополнительных отношениях f и g этой единственной вершины с другими. Продолжая аналогию между вершинами SDL и сети Петри можно ЗАДАЧЕ по-

ставить в соответствие переход сети. К сожалению, в SDL-диаграммах не всякая пара вершин СОСТОЯНИЕ или РЕШЕНИЕ *(S и R)* разделяется вершиной ЗАДАЧА, хотя смена любого состояния УК предполагает выполнение функции переходов и (или) выходов, а следовательно постановку некоторой задачи, в ряде случаев вырожденного характера. По правилам SDL два СОСТОЯНИЯ разделяются ПЕРЕХОДОМ, который обязательно включает один ВХОД и может включать любое количество ВЫХОДОВ, ЗАДАЧ и РЕШЕНИЙ. Между любыми двумя вершинами из множества (SuA) может иметься или нет один ВХОД, любое количество ВЫХОДОВ и ЗАДАЧ, или не иметься ни одной вершины. Поэтому при переходе к сетевому описанию каждая пара позиций, соответствующая последовательно принимаемым состояниям УК, разделяется одним переходом, даже если между соответствующими им символами SDL отсутствует ЗАДАЧА. Это отвечает трансляции исходящих и входящих дуг вершин множества SDL в переходы СП. Будем считать, что в этом случае переход сети соответствует мнимой ЗАДАЧЕ. ВХОД и ВЫХОДЫ, содержащиеся между двумя элементами множества отображаются соответственно входящими и исходящими дугами, связанными с внешней средой. В общем случае, элемент ЗАДАЧА можно игнорировать.

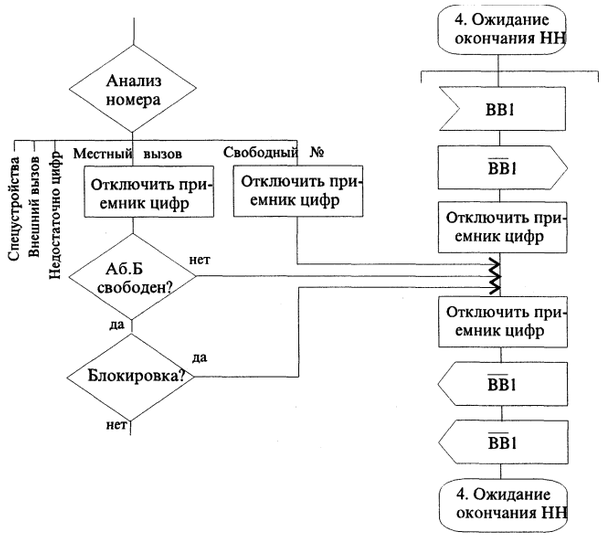


Рис.2.3 Пример узла в SDL-диграмме.

Возможны случаи, когда между двумя вершинами SDL из множества (5 и 7?) имеется узел из нескольких дуг, идущих от разных ветвей алгоритма. Фрагмент диаграммы с такой ситуацией приведен на рис.2.3 (взято из [7]). Возникает вопрос о месте включения дуг при сетевом описании процесса. Можно заметить, что входящие в пере-

ход дуги соединяют его с позициями, определяющими условия срабатывания перехода, а дуги входящие в вершину определяют возможность поступления заявки на некоторую фазу обслуживания. При этом поступления заявок из разных ветвей взаимонезависимы, а переход к этой фазе произойдет после одинаковых событий, но вызванных разными заявками (в примере — "установка сигнала S"). Такую логику процесса можно отразить сетью Петри, если параллельные ветви замкнуть на первую позицию — образ вершины SDL из множества (Su Я), следующую после узла, и все позиции разделить переходами. При этом переход дополняется всеми дугами — образами вершин ВХОД и ВЫХОД встреченными на пути между парой вершин из *(SuR)*. Фрагмент сети, соответствующий примеру на рис.2.3 показан на рис.2.4. В результате перехода к сетевому описанию образовано 5 позиций и 6 переходов, из которых, вместе с наборами дуг SDL, переходам dl, d4, d5 соответствует один символ ЗАДАЧА, переходам d3, d6 — два, переходу d2 — мнимая ЗАДАЧА. Дуги СП, соответствующие ВХОДАМ и ВЫХОДАМ должны быть соединены с позицией, которая моделирует окружение УК. Необходимо заметить, что условия срабатывания конфликтующих переходов, следующих после позиций, соответствующих РЕШЕНИЯМ, не определены и могут зависеть как от внешнего окружения (тип вызова, занятость абонента и т.д.) так и от наличия внутренних ресурсов (свободность ПНН, СЛ и т.п.). Вопрос доопределения условий их срабатывания рассмотрен в 2.2.

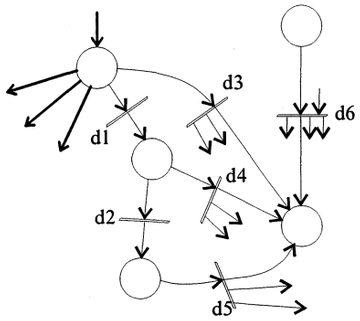


Рис.2.4 Фрагмент Сети Петри, соответствующий диаграмме на рис.2.3[1]

Как видно из примера, SDL-диаграмму можно рассматривать как совокупность вершин, соответствующих устойчивым и неустойчивым состояниям, все элементы соединяющие их — как переходы между ними, направленность процесса определяется направленностью дуг диаграммы.

[1] Здесь и далее фрагменты СП изображаются с разомкнутыми дугами для уменьшения затенения рисунков. Имеется ввиду, что инцидентные им элементы СП обязательно входят в состав полной СП.

Дадим формальное описание предлагаемой процедуры перехода от SDL-диаграммы к сети Петри. Введем обозначение *SR =S uR,* то есть

$$SR = \{sr_u\ sr_2, \ldots\ sr_{,,}\},$$

где и — натуральное.

Два элемента из этого множества sr_t и лт, будем считать *соседними,* при условии

$$((srj£\ \underbrace{f(f(f(}_{m\ раз}\ \ldots f\ (sri)\ldots)= /''(srj)\ \underbrace{v(sr_t\ eg(g(g(\ldots\ g(srj))\ldots)}_{m\ раз} = g^m(sr_j))A$$

$$л\ \overset{m-1}{\underset{k=\backslash}{}}\text{fry}\ /*\ (\overset{m-1}{\underset{k=4}{sr_,}})\ г\ SR)\ v(\backslash/S\ (srJeSR)),$$

где m — число дуг диаграммы между sr_t и sr_r

$i, j, к$ — индексные переменные, i, j, k, m — натуральные.

Выражение означает, что между двумя элементами из *SR* дуги одного направления, нет параллельных ветвей и других элементов из *SR.* Считаем такую пару соседних элементов связанной функциями *следования состояний F* и *предшествования состояний G:*

$$F(sr.) = f(sr,) = \{^\};\qquad (2.1)$$

$$СГ^=г''^ = \{^,\}.\qquad (2.2)$$

Пусть макропозиция *V* моделирует внешнее окружение УК и связана с множеством *D* функциями:

$$Ф: VxD—> \{0;1\};$$

$$H:DxV-» \{0, 1, 2,...\}.$$

Для перехода к сети Петри необходимо установить соответствия:

$$\underset{1=2}{\text{V}}\ sr, -> Ь,;\qquad (2.3)$$

$$(\underset{j=l}{\text{У}},\ ^{sr}j\epsilon\ ^{F}(^{sr}i)\bullet'\ \{^{sr}i\ '>^{sr}j\}\ _{э}\qquad \underset{1=2}{srj}\ G(srj);\ _{э}\{sr_l\ ;srj\ \}_{э}->d_k);\qquad (2.4)$$

$$Ф(Ь_i,a_к)=1;\qquad H(d_k,b_j)=l;\qquad (2.5)$$

$$Ф(a_к,У)=p;\qquad H(d_k,V)=q,\qquad (2.6)$$

где p — число ВХОДОВ I между sn и srj, $p\ \epsilon\ \{0;1\}$

q — число ВЬПСОДОВ О между и srj, $q\ \epsilon\ \{0; 1; 2;...\}$

$$d=f^*(sr,))\ _V(i =_k\text{-}'(sr_t));\ x\epsilon\{1,2...m\}\qquad (2.7)$$

$$(O =f'(sr,))v(O =g^r(sr_J));\ y\ \epsilon\{1,2...m\}\qquad (2.8)$$

Обычно SDL-диаграммы начинаются вершиной типа СОСТОЯНИЕ — "Ожидание вызова", “Свободно” и т.п. Соответствующая ей позиция СП не может опи-

сывать фазу обслуживания заявок, в отличие от других, поскольку нет еще самой заявки. Смысл вершины — готовность устройств УК — говорит о свободное™ ресурсов, захватываемых заявкой при поступлении (например память массива обслуживаемых вызовов). Поэтому в формуле (2.3) учтено, что начальная вершина SDL (“Ожидание вызова”) не отражает стадии обслуживания заявки (z=2).

Сеть, полученная в результате данной процедуры, описывает фазы обслуживания вызова, требуемую логику взаимодействия с окружением, и представляет собой часть моделирующего алгоритма, которая определяется составом и порядком выполнения УК требуемых функций. Поэтому, ее топология не зависит от конкретной реализации УК — степени централизации управления, емкости и организации памяти, быстродействия и т.д., и как следствие “автоматное™” базовой модели SDL, она является автоматной по построению. Описание ресурсных отношений рассмотрено в п.2.3. В такой модели реального УК необходимо отразить также временные свойства сети введением временных задержек, которые определяются временем выполнения частных алгоритмов, описываемых переходами СП (рассмотрено в п.2.1.2). Кроме того, очевидно необходимо моделирование потока заявок, поступающих в УК из внешнего окружения, которое условно обозначено как макропозиция V (рассмотрено в п.2.2). Поэтому будем рассматривать общую сетевую модель, как совокупность трех основных подсетей:

- алгоритма,
- внешнего окружения,
- ресурсов.

Как было указано выше, подсеть алгоритма, полученная в результате описанной процедуры перехода от SDL, является недетерминированной, т.е. содержит переходы конкурирующие за метку, последовательность срабатывания которых не определена (например dl и d3, d2 и d4 на рис.2.4). Доопределение таких переходов зависит от содержательного смысла позиций и производится входящими дугами либо из внешнего окружения, либо от некоторых позиций, отражающих состояние тех или иных объектов УК, интерпретируемых как ресурсы.

Отметим некоторые очевидные свойства подсети алгоритма, как графа СП:

- подсеть является автоматной [34];
- подсеть разомкнута [66], то есть содержит терминальные элементы;

- подсеть может включать локальные циклы (замкнутые маршруты) для повторяющихся процедур, например прием цифр номера.

Подведем краткое резюме действий по переходу от SDL диаграммы к СП подсети алгоритма:

1. По элементам множества *SR* образовать позиции (выражение (2.3));
2. По отношениям предшествования состояний и следования состояний (выражения (2.1) и (2.2)) образовать переходы сети (выражение (2.4)) и установить между ними отношения инцидентности (выражения (2.5) и (2.6));
3. По вершинам ВХОД и ВЫХОД находящимся между соседними вершинами *sr* образовать дуги к подсети внешнего окружения;
4. Присвоить элементам сети временные задержки[1].

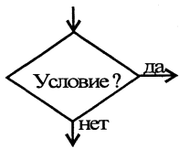
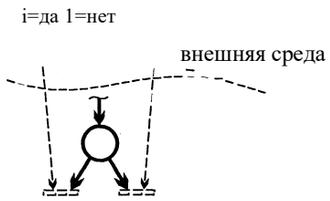
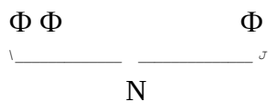
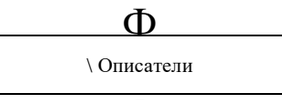
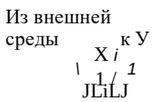
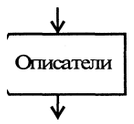
В табл.2.1.1 представлены графические соответствия элементов SDL-диаграммы и графа СП подсети алгоритма, которые удобно использовать при трансляции SDL-диаграммы в СП[2].

Таблица 2.1

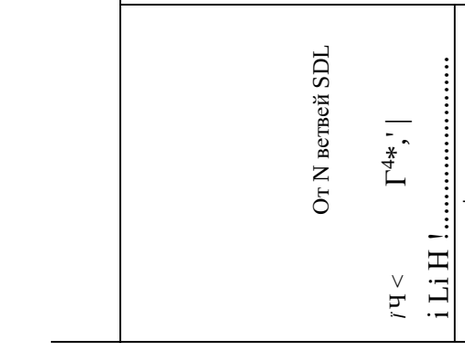
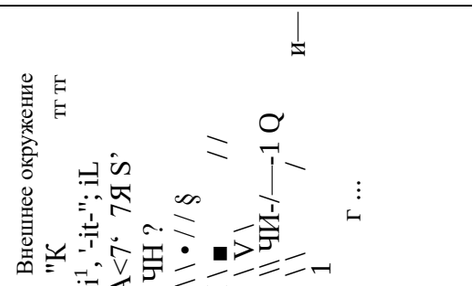
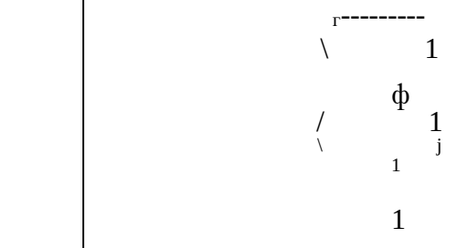
№ п/п	Форма языка SDL	Форма сети Петри
	Словесное описание	Словесное описание
	Графическое изображение	Графическое изображение
1.	Любая вершина “Состояние”, исключая начальную.	Вершина-позиция
	Ф **№ Описатель^** **4**	
2.	Начальная вершина “Состояние” *(“Ожидание вызова ”, “Свободно ”* и т.п.)	В элементы СП не транслируется
	Ф **0. Ожидание I** **вызова)** **ф**	***Игнорировать***

[1] Вопрос о том, каким именно элементам присваивать задержки — позициям или переходам, рассмотрен в общей концепции модели в 4.2.1.
[2] Вершины “Задача” и “Запоминание” SDL могут просто игнорироваться, иначе они совмещаются с совокупностью дуг SDL — п.9 таблицы 2.1.

Продолжение табл. 2.1

	Любая вершина “Решение'	Вершина-позиция
		
4.	Вершина “Решение” с последующей расходимостью	Вершина-позиция с прямыми инцидентностями
		
5.	Любая вершина “Вход”	Инцидентная дуга из внешнего окружения
		
6.	Любая вершина “Выход”	Инцидентная дуга во внешнее окружение
		
7.	Любая вершина “Задача” совместно с совокупностью дуг — п.9 таблицы	Вершина-переход (или игнорировать)
		

Продолжение табл. 2.1

8.	Сходимость дуг SDL	Отдельные переходы, инцидентные одной позиции
		
9.	Совокупность дуг одного направления между парой “Состояний” или “Решений”	Отдельный переход, инцидентный паре соответствующих позиций
		
10.	Любая вершина “Запоминание” — аналогично п.7 таблицы “Задача”	Вершина-переход (или игнорировать)
		
11.	Символ “Стоп”	Терминальный переход

Пример перехода к сетевому описанию SDL-диаграммы на рис.2.1 по действиям 1-3 приведен на рис.2.5. Штриховой линией показано доопределение переходов подсетью ресурсов и пример указания ресурса — “количество ПТН”. Пучки дуг связанных с одной вершиной показаны на рисунке сливающимися линиями для уменьшения затемненное™ чертежа большим количеством инцидентных дуг.

В заключение отметим, что сетевое описание полученное на основе SDL-диаграмм является лишь верхним уровнем иерархии моделирования УК. Каждую позицию или переход можно представить подсетью более низкого уровня иерархии, переходя, например, к частным алгоритмам (приема сигналов вызова, определения промпути и т.д.). Такой переход можно использовать для получения временных параметров элементов СП, параметров случайных законов для моделирования и т.д. (см п. 2.1.2), либо прямо включать в общую сетевую модель. В последнем случае, очевидно, происходит усложнение модели. Возможны приемы и для ее упрощения, например если приемлемо предположение о том, что длительность формирования внешнего сигнала значительно меньше времени реакции абонента, то описание его выдачи (инцидентную дугу СП к макропозиции V) можно опустить.

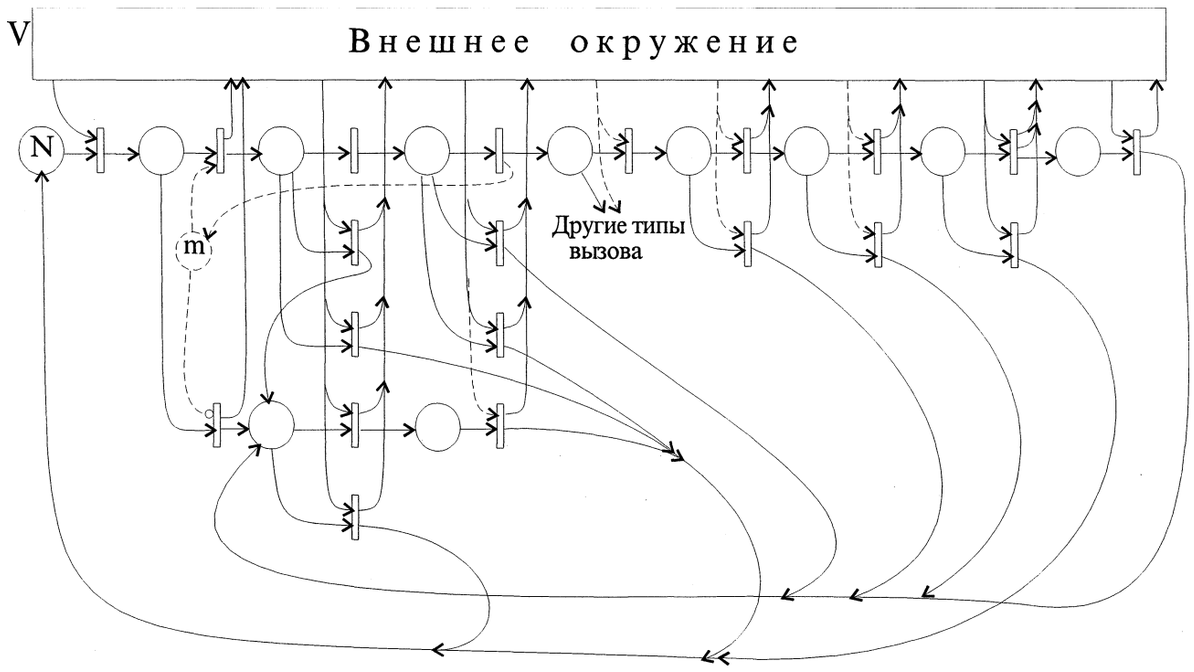


Рис.2.5. Сетевая модель построенная по SDL-диаграмме на рис.2.1:
N - количество меток отражающее производительность управляющего устрой-
ства или емкость массива обслуживаемых вызовов, ш - количество

2.1.2. Частные алгоритмы работы управляющих устройств узлов коммутации[1]

Как было указано выше, описание общего алгоритма функционирования УК целесообразно производить на этапах спецификации-планирования и системного проектирования, когда определяется состав входных, выходных сигналов, состояний УК, его структурная схема. Более подробное описание отдельных объектов (модулей) как аппаратных, так и программных рекомендовано на этапах детального (технического) проектирования, программирования. Однако для удовлетворительной достоверности выбора параметров общей сетевой модели, определения степени различия возможных вариантов, необходима оценка временных и количественных характеристик отдельных объектов УК. Поэтому можно считать целесообразным предварительное функциональное описание ряда частных алгоритмов, существенно влияющих на выбор временных задержек переходов кратностей дуг и начальной разметки общей СП.

Такому подходу способствует тот факт, что в большинстве случаев, частные алгоритмы уже разработаны, модификациям подвержены умеренно, так как совершенствование УУ идет в основном по пути создания более эффективной, быстродействующей элементной базы, улучшения (интенсификации) использования памяти, модернизации операционных систем (ОС), оптимизации общих структур УУ УК. Кроме того, СП становятся все более популярными, как средство разработки, анализа и оптимизации алгоритмов программ. Таким образом появляется возможность построения сетевой модели частного алгоритма по разработанному ранее алгоритму программы с последующими испытаниями ее для получения указанных выше параметров.

В п.2.1.1 уже упоминалось, что блок-схемы программ легко транслировать в СП [59]. При этом вершинам блок-схемы соответствуют вершины СП, а дугам — инцидентные им переходы. Пример такого преобразования представлен на рис.2.6.

Перемещение единственной метки в СП будет соответствовать ходу выполнения реальной программы, если переходам присвоить временные задержки, соответствующие быстродействию вычислительного устройства и числу выполняемых микроопераций в соответствующем операторе, а порядок срабатывания конфликтующих переходов (на рис.2.6 — (ds ds) и (dg d?)) — порядку передачи управления в программе.

---

[1] В данном пункте использованы некоторые решения по общей концепции сетевой модели — см. п. 4.2.1.

Отсюда принципы построения модели в терминах принятого расширения СП:

1. По описанному правилу построить граф СП.
2. Назначить временные задержки переходов (в общем случае могут быть случайными).
3. Головную позицию обозначить как генераторы случайных чисел (ГСЧ)[1] с соответствующим законом распределения.

4.0бщие для конфликтных переходов позиции обозначить как “ждущие ГСЧ” (ЖГСЧ)[2 3] с соответствующими законами распределения.

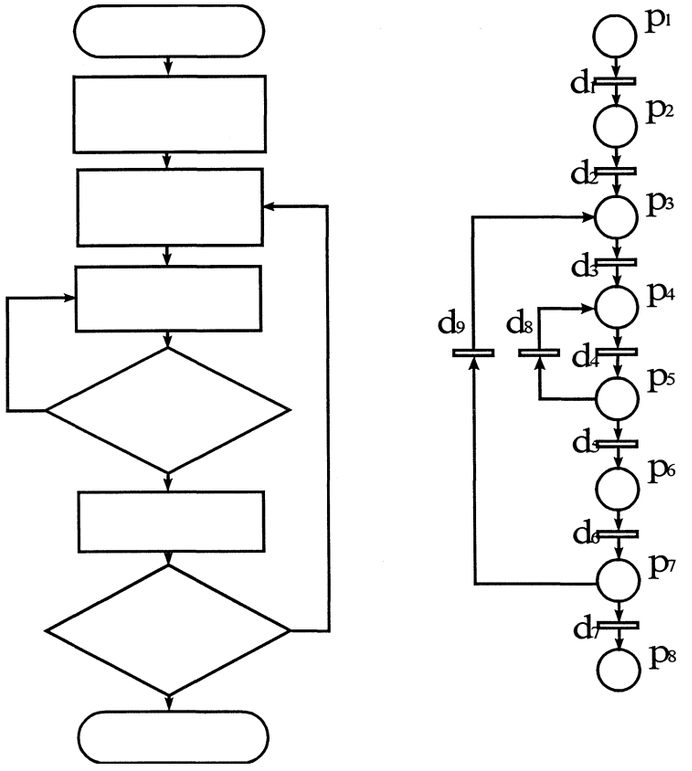
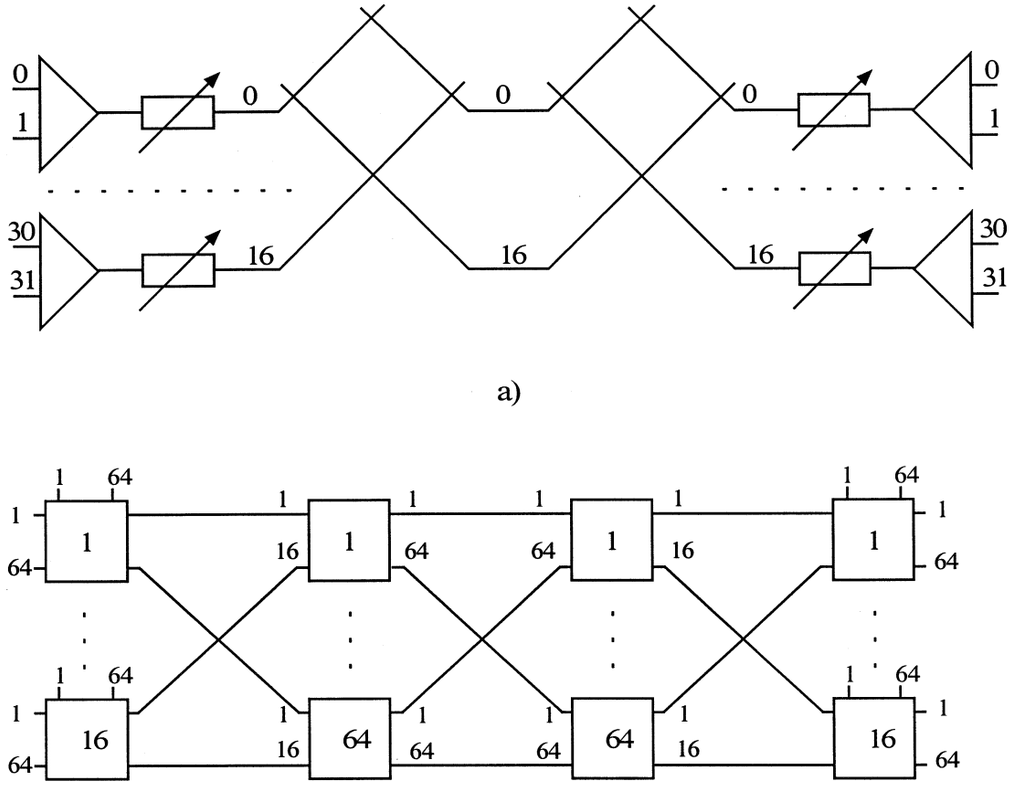


Рис.2.6. Преобразование блок-схемы программы в СП

Такая модель уже способна описывать одновременное обслуживание нескольких заявок в многопрограммном режиме. Анализ ее на временные свойства позволит получить временные параметры по выполнению программы (среднее время выполнения, функция распределения времени выполнения) в условиях одновременного обслуживания случайного потока заявок. В качестве примера рассмотрим вариант алгоритма обусловленного поиска промежуточного пути в четырехзвенной КС структуры В-П-П-В (рис.2.7.а)[3]. Такую КС можно представить в виде пространственного эквивалента

[1] См. подраздел 2.3
[2] См. подраздел 2.3
[3] упрощенно соответствует МТ-20/25

(рис.2.7.б). Данная коммутационная структура имеет параллельно-последовательный граф доступности, показанный на рис.2.8.



а)

Рис.2.7. Четырехзвенная коммутационная система

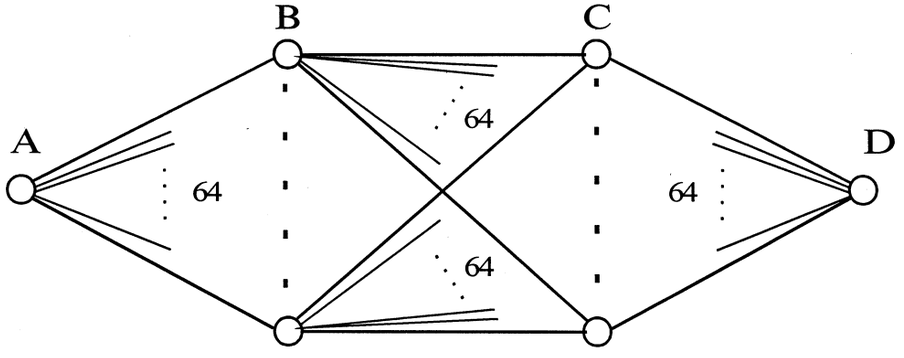


Рис.2.8. Граф доступности четырехзвенной КС

Этому графу доступности соответствует длина слова состояния выходов любого звена — 64 бит. Алгоритм поиска промпути, при условии того, что соединения двухсторонние, используется размножение состояний и простейшая процедура выбора первого найденного свободного промпути, представлена на рис.2.9.

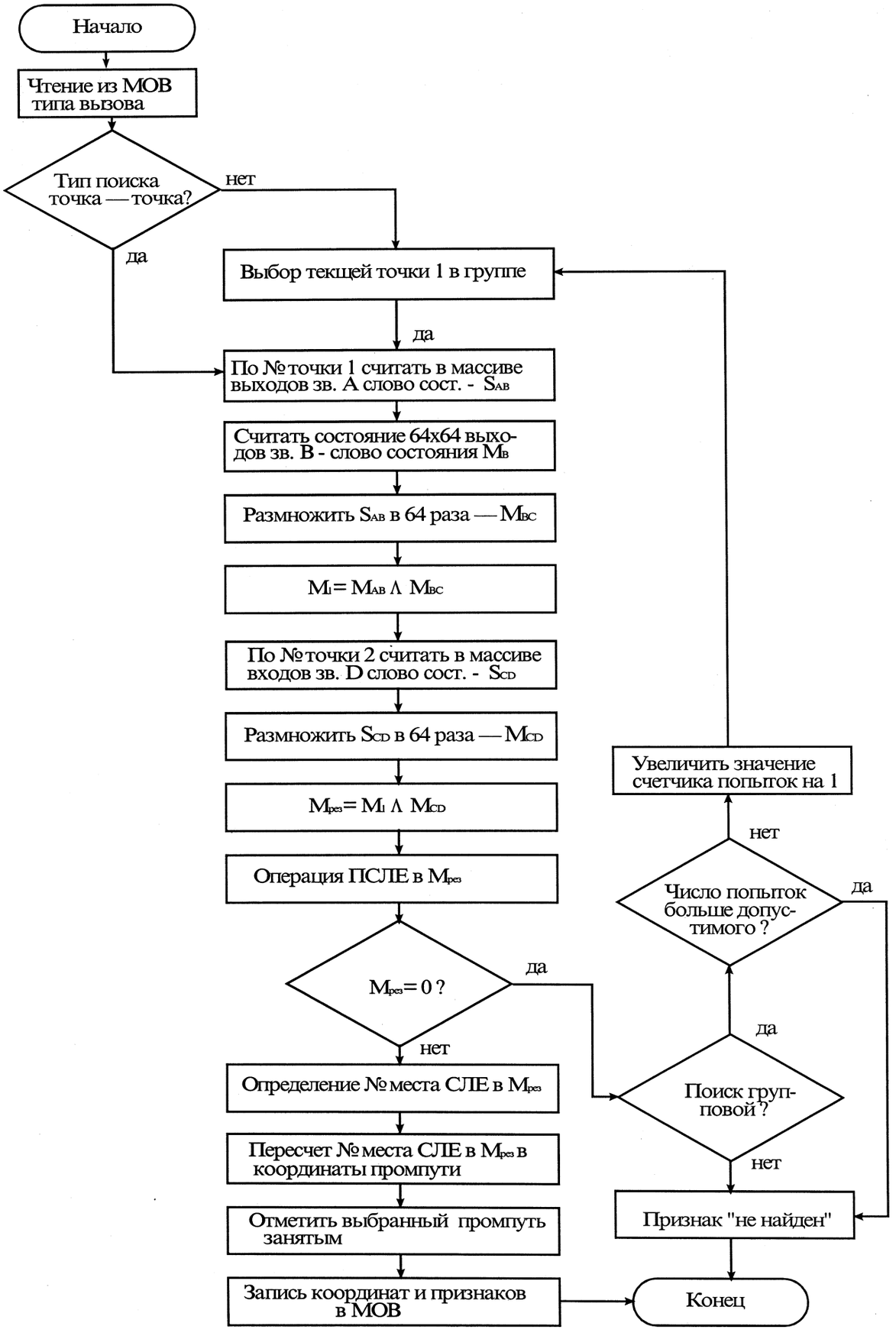


Рис.2.9. Алгоритм поиска промпути в четырехзвенной КС

В соответствии с указанным выше правилом строим СП (рис.2.10)

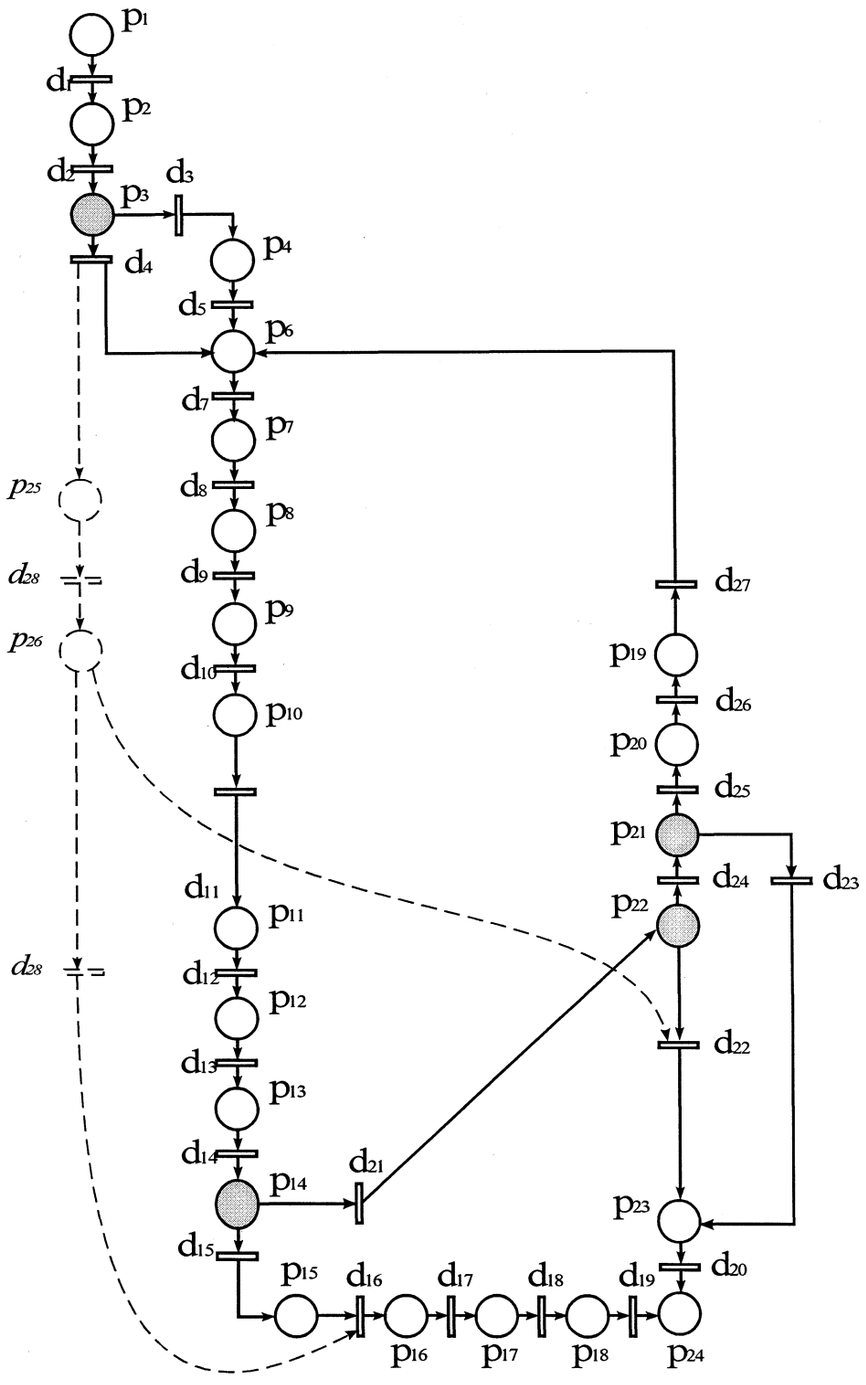


Рис.2.10. Сетевая модель поиска промпути

Временная задержка di определяется длительностью выполнения операций ОС по вызову (активизации) данной подпрограммы, а остальных переходов — назначается в соответствии с составом и длительностью выполнения микроопераций по соответствующей операции алгоритма. Вероятности по выходам p3 и p4 — определяются данными о спектре заявок, pid — вероятностью блокировки в КС данного типа для потока

заявок создаваемого ГСЧ pi, P22 — совпадает с pз (считаем, что вероятность внутренней блокировки не зависит от типа поиска), prі — вероятностью превышения заданного числа попыток при условии, вызов групповой и предыдущая попытка была неудачной. Рассмотрим пример задачи определения вероятностей по выходам prі и P22. Для этого рассмотрим места разветвления алгоритма в следующей постановке.

Дано: На вход устройства поступают заявки, которые обслуживаются в соответствии с графом (рис.2.11):

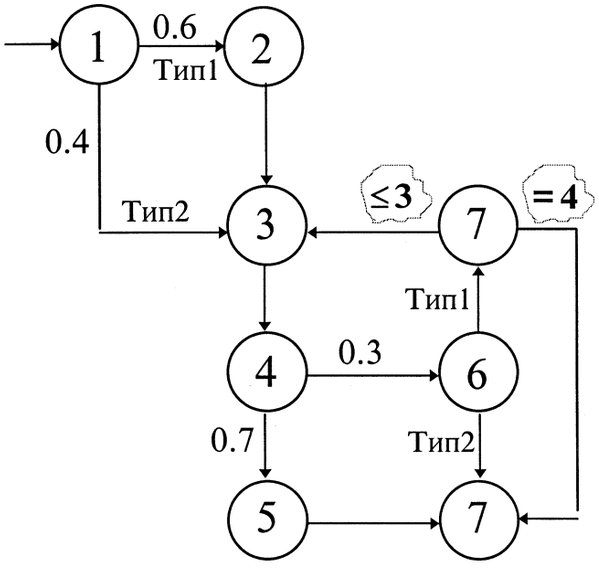


Рис.2.11. Граф обслуживания заявок к примеру задачи определения вероятностей

Вершины соответствуют действиям, производимым с заявкой, которые зависят от ее содержания:

-в вершине 1 определяется тип: с вероятностью p=0,6 — тип 1, p=0,4 — тип 2;

-в вершине 4 с заявками любого типа производится действие, по которому с вероятностью p=0,3 она попадает в вершину 6, с p=0,7 — в вершину 8;

-в вершине 6 вторично определяется тип, и, в зависимости от результата, обслуживание заканчивается (вершина 8) или повторяется (вершины 3, 4), если число попыток не более 3;

— в вершине 7 определяется число попыток по данной заявке, к числу прибавляется 1, и если оно больше трех — обслуживание заканчивается.

Определить:

1. С какой вероятностью заявка, попавшая в вершину 6 имеет тип 1?

2. С какой вероятностью обслуживание заявки, попавшей в вершину 7, завершается?

Решение:

1. Определим части заявок типа 1, попадающих в вершины 6 и 7, по отношению к общему числу заявок , поступивших извне:

1-кратно обслуженных: vi' = 0,6 • 0,3 = 0,18

2-кратно обслуженных: V2' = 0,18 • 0,3 = 0,054

3-кратно обслуженных: vV — 0,054 • 0,3 = 0,0162

2. Определим часть заявок типа 2, попадающих в вершину 6, по отношению к поступившим извне заявкам:

vi" = 0,4-0,3 = 0,12

3. Вероятность того, что заявка в вершине 6 типа 1, определим как отношение соответствующих частот:

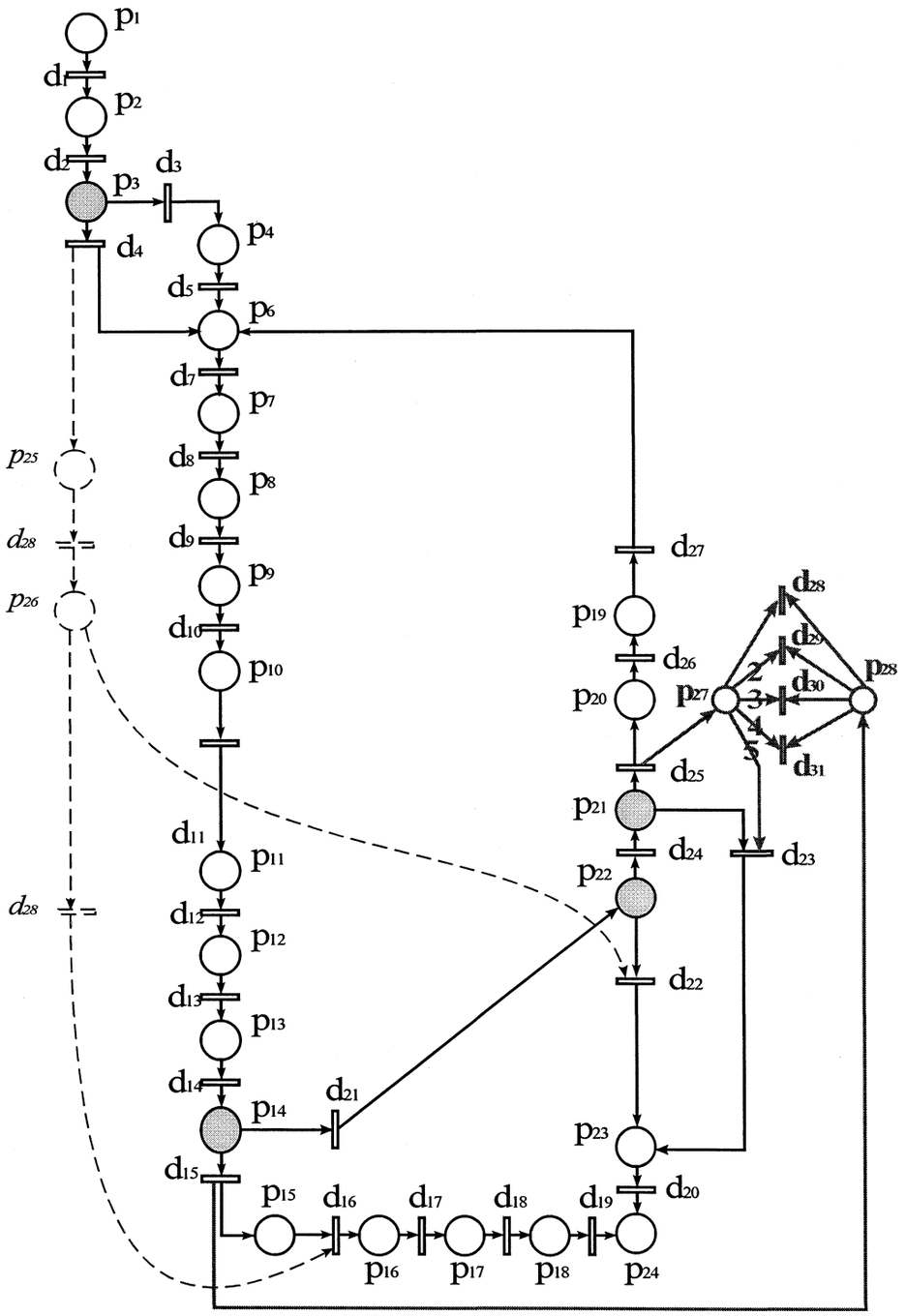
$$P\backslash \quad \frac{\nu_1'+\nu_2'+\nu_3'}{\nu_1'+\nu_2'+\nu_3'+\nu_1''} \quad \frac{0{,}18 + 0{,}0162 + 0{,}054}{0.18 + 0.054 + 0.0162 + 0.12} = \frac{0{,}2502}{0.3702} = 0{,}67585$$

4. Вероятность завершения обслуживания (число попыток уже 3):

$$P2 \quad \frac{\hat{3}'}{{}^{V}1^{/+V}2^{Z+V}3^{/}} \quad \frac{0{,}0162}{0{,}18 + 0{,}054 + 0{,}0162} = 0{,}6475$$

Заметим, что случайное распределение меток в узловых позициях описывает алгоритм в наиболее общем виде и не "отслеживает" перемещение каждой заявки, что не позволяет получить некоторые усредненные параметры. Однако в ряде частных случаев такое вероятностное описание (ЖГСЧ) можно заменить детерминированным (сетевым). Например, если считать, что временные задержки переходов постоянны, тогда повторяющийся условный оператор, который проверяет условие, уже проверенное ранее, можно транслировать не в ЖГСЧ, а в обычную позицию, общую для 2—х конфликтных переходов (на рис.2.10 — Р22 и 622, бгз). Конфликт последних устраняется введением дополнительной цепи р25— 628—Р26—<I29(cI22) (рис.2.12) и установкой приоритетов: pr(d22)>pr(d24)^pr(d29). В р25 отмечается факт принадлежности поиска к типу "точка-точка", d28 моделирует временную задержку обработки заявки, а 629 — отсутствие необходимости различения типа заявки при успешной попытке поиска ("очистка" ргб). Поскольку сеть работает в синхронном режиме, то в момент появления метки в ргб, — в позиции р22 либо имеется метка, соответствующая данной заявке (при неудачной попытке поиска), либо нет ни одной метки (при удачной попытке). В первом случае срабатывает

d22, во втором — d29. Дополнительная цепочка элементов СП как бы "отслеживает" возможное попадание заявки на поиск типа "точка-точка" в число не обслуженных (ргг).

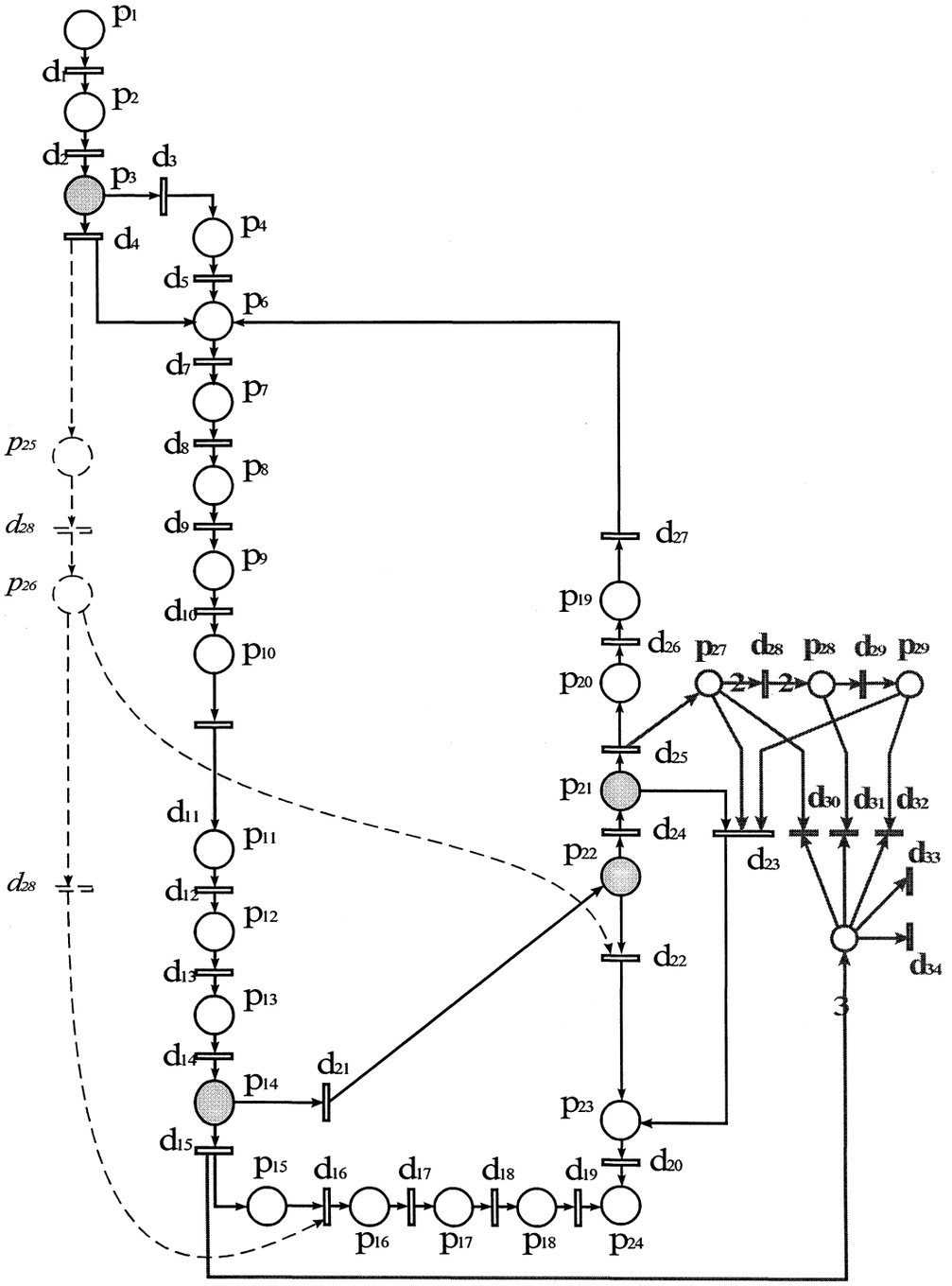


РгССЬ) > Рг(бзі) > Рг(ёзо) > Рг(сЬ) > Рг(d>$_8$)

Рис.2.12.а. Описание счетчика попыток обслуживания

В другом частном случае, при приемлемости предположения о том, что очередная заявка не принимается на обслуживание, пока не закончена обработка предыдущей,

можно заменить ЖГСЧ — образ условного оператора сравнения числа попыток с допустимым, — более точным сетевым описанием этого оператора. В соответствии с логикой работы оператора, СП должна содержать счетную подсеть, запоминающую число попыток (см. п.4.2.2); подсеть, восстанавливающую ее исходное состояние (разметку) в случае удачной попытки, или превышения заданного числа попыток.



Рг(сЬз) > Рг(<±о) > Рг(<±!) > Рг(<±2) > Рг(с3зз) > Рг(<±4) > Рг(d2s) > Рг(сЬ)

Рис.2.12.6. Описание счетчика попыток обслуживания

Возможные варианты модели, при числе попыток п<5, представлены на рис.2.12а,б и различаются построением счетных подсетей. Заметим, что счетная сеть на рис.2.12б при $п>2^4$ содержит значительно меньшее число переходов и позиций, чем на рис.2.12а.

Обобщим действия по построению сетевых моделей частных алгоритмов функционирования УК следующей последовательностью шагов.

1. Преобразовать блок-схему алгоритма в СП, заменив вершины блок-схемы позициями, а дуги — переходами. Отношения инцидентности установить соответственно направлению дуг блок-схемы.

2. Назначить временные задержки переходов как сумму длительностей выполнения микроопераций в операторе, соответствующем его прямо инцидентной позиции.

3. При необходимости назначения случайной временной задержки использовать методику, приведенную в п.2.2 (ЖГСЧ).

4. Головную позицию обозначить как ГСЧ, устанавливающий закон распределения последующих заявок на алгоритм.

5. Позиции, соответствующие условным операторам обозначить как ЖГСЧ, установить закон распределения меток по выходам на основе статистики (спектра занятий и др.) или расчетом[1].

6. Если имеются данные об ограничениях на временной режим или дисциплину обслуживания по данному алгоритму, позволяющие заменить ЖГСЧ детерминированными фрагментами СП — произвести замену.

7. В полученной модели для определения искомых временных характеристик образовать дополнительные позиции и переходы (см. п.4.2.2).

[1] Методика расчета вероятностей в подобных задачах изложена в [22].

2.2 Сетевое описание внешней среды узлов коммутации

Одной из проблем, возникающих при моделировании УК, является адекватность отображения совокупности сигналов, поступающих из внешнего окружения (ВО) и отражающих воздействие на УК абонентов и сопряженных УК. Распространенным подходом в моделировании окружения УК является использование известных моделей потока вызовов для различных ступеней искания и этапов обслуживания, причем параметры потоков задаются на основании статистических данных. Примеры — простейший поток для описания поступающих вызовов и сглаженный поток для заявок на выходе i-й ступени искания (i = 1, 2,...) [33]. Такой подход удобно использовать при раздельном анализе пропускной способности частей оборудования УК, например ступени ГИ АТС координатной системы. Системный подход предполагает изучение процессов на уровне всей системы. Поэтому, вместо отдельных автономных потоков заявок, каждый из которых используется для моделирования части УК необходимо иметь систему увязанных потоков, которые отражают как статистические свойства ВО, так и воздействия самого УК на формирование потоков заявок. В общем контексте сетевой модели УК данная функция выполняется подсетью ВО.

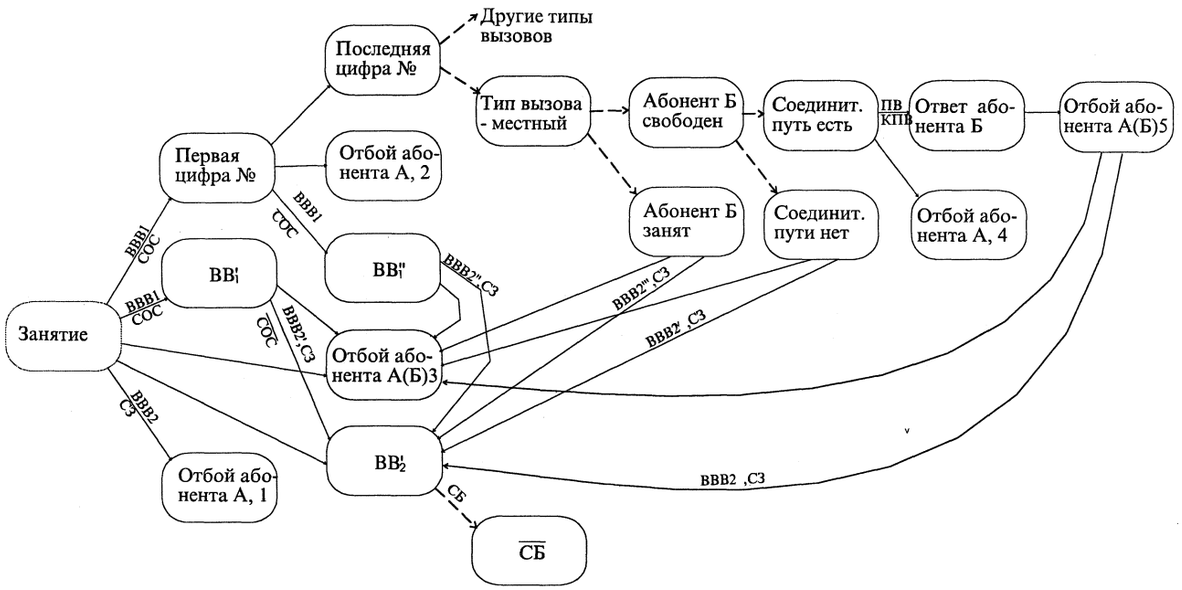


Рис.2.13. Граф смены сигналов внешнего окружения

Будем понимать под *первичными* заявками такие, которые вызывают постановку вызова на обслуживание: занятие АЛ и СЛ; поступление заголовка первого пакета в режиме виртуального канала (ВК), поступление датаграммы и т.п. Под *вторичной* заявкой будем понимать заявку, поступление которой возможно лишь при условии того, что соответствующий вызов (первичная заявка) уже находится на обслуживании: набор номера, отбой на различных стадиях установления соединения, тип вызова, выдержка времени, поступление заголовков последующих пакетов в режиме ВК и т.п. При использовании для представления каждого вида заявок автономных генераторов случайных чисел теряется взаимосвязь между моментами поступления первичных и вторичных заявок, что может привести, например, к поступлению отбоя до занятия, что неприемлемо для анализа работы УК как единой системы.

Более адекватную модель внешних воздействий можно получить если в качестве источника первичных заявок использовать одну из известных математических моделей (простейший, примитивный, с повторными вызовами и другие потоки), а вторичные потоки формировать из первичного последующими модификациями учитывающими как статистические сведения об окружении, таки воздействие сигналов от самого УК во внешнюю среду. Как было указано выше, для представления сложных моделей удобно использовать аппарат сетей Петри.

Для построения такой сети, совокупность заявок из ВО и логику их смены удобно представить в виде *графа смены сигналов* на основе анализа SDL-диаграммы. Пример такого графа, соответствующего фрагменту SDL-диаграммы из п.2.1.1, показан на рис.2.13. Здесь вторичным заявкам соответствуют вершины ВХОД и РЕШЕНИЕ SDL-диаграммы. На рисунке показаны так же сигналы вырабатываемые УК и их связь со сменой внешних сигналов. Граф смены сигналов несложно построить непосредственно по SDL-диаграмме или по модели работы УК в виде подсети алгоритма (п.2.1.1).

Для перехода к СП поток первичных заявок можно представить как поток меток, вырабатываемых генератором случайных чисел, а потоки вторичных заявок — разделением первичного потока при помощи "просеивания". Для отражения случайной природы поступления заявок необходимо задание вероятностных распределений для "просеивания" и случайных временных в соответствии со статистическими сведениями, известными для каждого типа вторичных заявок. Кроме того, условия срабатывания

некоторых переходов должны быть дополнены получением соответствующих сигналов от самого УК, например первая цифра номера не может поступить до выдачи сигнала ответа станции, а отсчет выдержки времени должен начаться с момента ее включения на УК. Имея ввиду что событиям во внешнем окружении УК соответствуют переходы СП, а условиям их поступления — позиции сети можно совершить переход от графа на рис.2.13 к некоторому расширению СП, если вершинам графа поставить в соответствие позиции сети, а пучкам дуг из одной вершины — переходы (рис.2.14). Предполагается специальная логика работы переходов такой сети для выполнения следующих операций с потоками меток:

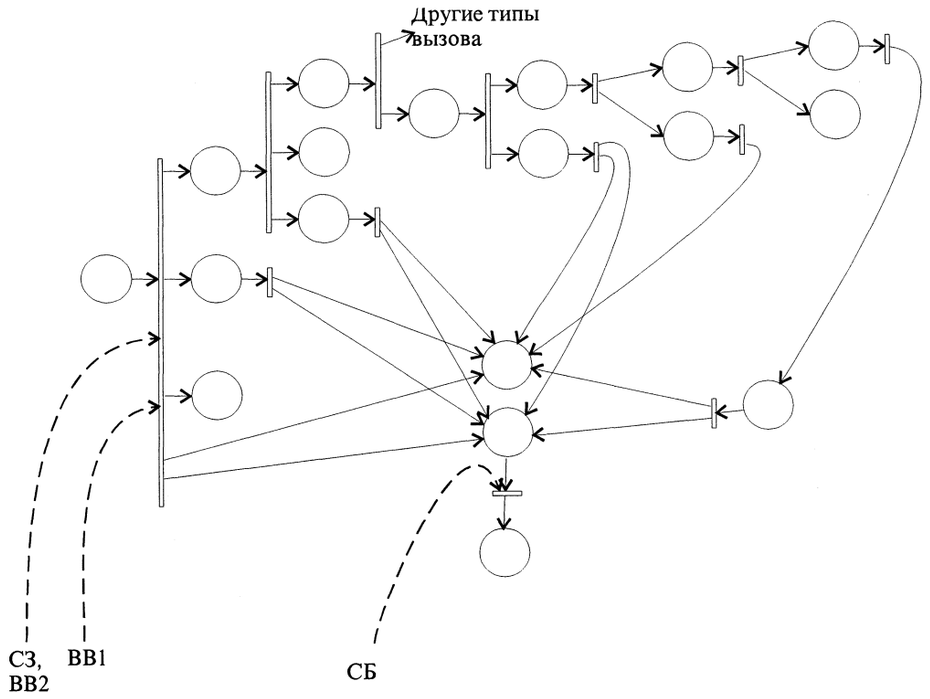


Рис.2.14 Первичная сеть по графу на рис.2.13

- установление случайных временных задержек по определенному закону распределения для каждого потока
- разделение потоков задержанных вторичных заявок по признаку типа заявки;

Задать параметры случайных законов для формирования потоков заявок в ВО обычными временными сетями невозможно, что требует их расширения введением случайной функции генератора случайных чисел (ГСЧ) которому в графе соответствует некоторая позиция, объявленная как генератор случайных меток (ГСМ). Особенностью рассматриваемой сети является то, что ГСЧ, моделирующий поток вторичных заявок,

должен вырабатывать число не самостоятельно, а лишь при поступлении очередной метки в позицию соответствующую данной фазе обслуживания. Это соответствует включению отсчета случайной временной задержки по данной заявке. Будем называть такой ГСЧ "ждущим" — ЖГСЧ, а соответствующую позицию — ЖГСМ. Такой ЖГСМ должен иметь возможность получения очередной метки до выдачи предыдущих (одной или нескольких) через отрезок времени, определяемый присвоенным ГСЧ законом распределения. Возможная реализация такого макроперехода сети изображена на рис.2.15.

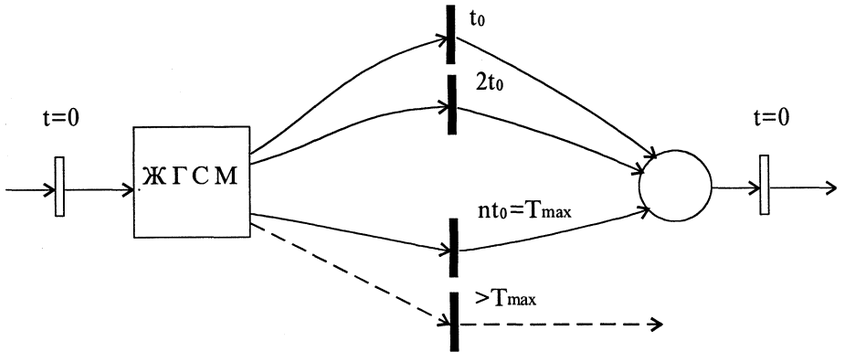


Рис.2.15. Моделирование случайной временной задержки ЖГСМ

Время t_0 представляет собой шаг нарастания временной базы и определяется требуемой точностью моделирования. Время T_{max} определяется выдержкой времени УК на данной фазе обслуживания или предельным значением времени задержки, вероятность которого пренебрежимо мала. ЖГСМ работает по принципу: поглощение одной метки на входе вызывает появление одной метки на выходе, номер которого определяется случайным числом ГСЧ. Для разделения потоков меток можно использовать такие же ЖГСМ, имеющие число выходов равное числу образуемых подпотоков. При необходимости ЖГСЧ может содержать выход "> Тмах 5 на котором появляются все метки, имеющие время задержки больше T_{max}, например для моделирования ситуации, когда в течение T_{max} не поступает цифра номера и в УК срабатывает выдержка времени ВВI. Как видно такое понятие является некоторой разновидностью управляющей функции, устанавливающей специальные правила (приоритет) возбуждения переходов. Логику работы временных переходов с различными временными задержками подсети ВО, как и других подсетей, можно представить в виде фрагмента СП, функционирующего по общим правилам временных СП (рис.2.16). Сумма задержек At дает время задержки перехода. Иногда случайные числа, вырабатываемые ГСЧ желательно представить в виде соответствующего количества порождаемых меток. Тогда логику работы "ждущего" ГСЧ

можно представить в виде ингибиторной сети Петри [34,59] (рис.2.17). Позиция G умножает поступившую метку в случайное число раз. Если это число менее 3, возбуждается один из переходов d2, d4, d6 в зависимости от значения случайного числа (0,1,2,...). В противном случае срабатывает переход d8, и оставшееся количество меток поглощается переходом d9. Переход d 10 "очищает" позицию p5 в конце цикла работы сети подготавливая ее к обработке новых меток. Заметим, что задержки переходов этой сети должны быть нулевыми, так как она реализует только "распределительную", а не "временную" функцию ЖГСЧ.

Рис.2.16. Вариант реализации макроперехода с временной задержкой и соблюдением очередности обработки меток

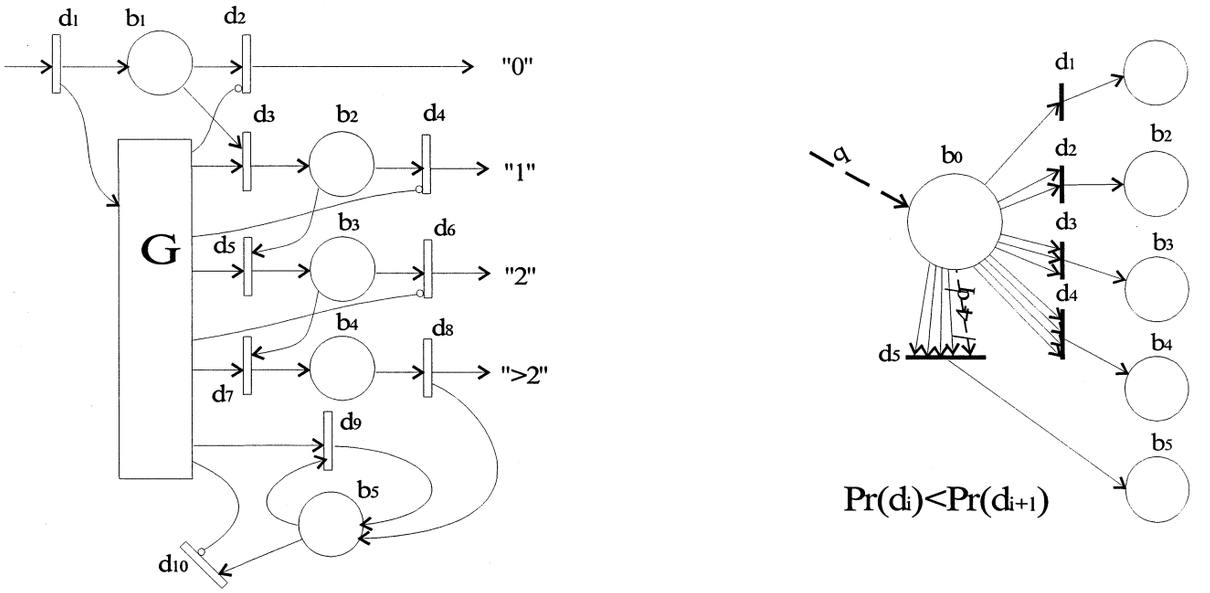


Рис.2.17. Варианты реализации макропозиции генератора вторичных заявок

Такая сеть эквивалентна приоритетной сети со случайной кратностью инцидентных дуг, как показано на том же рисунке (q — случайное целое положительное число).

Особенным случаем является наличие в подсети алгоритма локальных циклов, при повторяющихся процедурах обслуживания заявок, — обычно прием цифр номера в УК с коммутацией каналов. Подсеть ВО отражает действия абонентов по правилам взаимодействия с УК, в которые входит ожидаемое число цифр номера, определяемое нумерацией УК. Поэтому в посети ВО должен быть счетчик набранных цифр. Для описания

ситуаций преждевременного отбоя и длительного бездействия, позиции счетчика необходимо объявить ЖГСМ. На рис.2.18 представлен пример СП, построенной по фрагменту SDL - диаграммы из [42,66], поясняющий данное замечание[1].

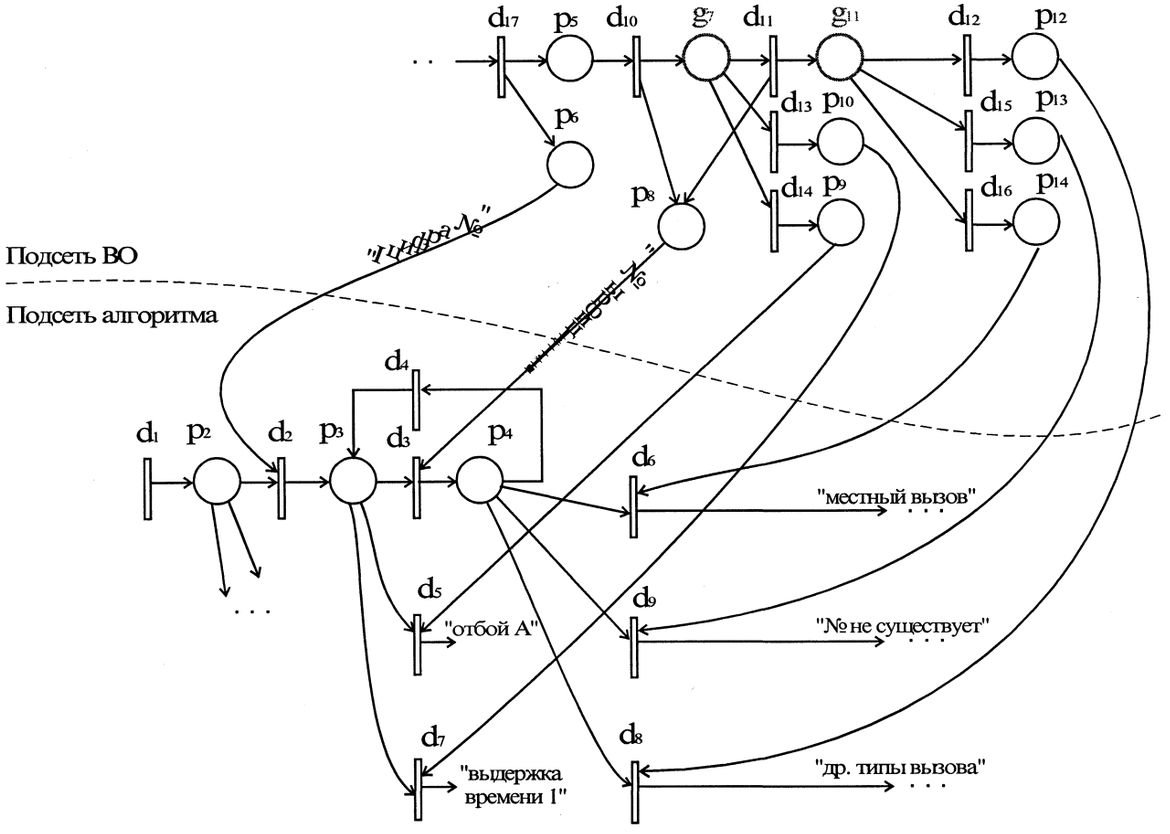


Рис.2.18. Описание ВО при наличии локального цикла в SDL-диаграмме

Используя описанные представления переходов и позиций первоначальной сети (рис.2.14) можно перейти к сети, функционирующей почти по правилам временных приоритетных или временных ингибиторных сетей и отличающейся необходимыми позициями ГСЧ и ЖГСМ. На рис.2.18 показан пример такой сети построенной по первоначальной сети рассмотренной выше. Точечной линией показано поступление меток соответствующих сигналам из УК во внешнюю среду, штриховой линией показаны элементы сети для передачи меток — вторичных заявок из ВО в УК. Переходы с временной задержкой закрашены. К подсети ВО предъявляются правила стыковки с подсетью алгоритма работы УК. Поскольку в 2.1 ВО описывалось некоторой макропозицией V, это означает, что в корректной подсети ВО все прямые маршруты (пути) к подсети алгоритма должны начинаться, а обратные — оканчиваться вершинами-позициями.

[1] Для простоты в примере не учитывается случайный характер времени поступления цифр номера.

Рассматриваемая разновидность СП близка к типу временных сетей N, формальное задание которых есть семерка множеств [59,62,66]:

N = { B, D, Ф, H, Mo, v,v},

где B — множество позиций, B ф 0;

D — множество переходов, D 0;

Ф — прямая функция инцидентности;

H — обратная функция инцидентности:

Ф : B x D-> { 0, 1 }; H : D x B-» { 0, 1 };

Mo — начальная разметка сети: Mo: B -> { 0, 1, 2,...};

o — временная база сети: и = {tl, t2,...};

v — функция временных задержек: v : D x и —> и.

Рассматриваемое расширение сети предполагает специальное подмножество позиций ГСМ BG И ЖГСМ BGG С соответствующими законами распределения:

BG C B, BGG <= B.

Стохастическая природа процессов в ВО, отражается случайной функцией (Ж)ГСЧ, которая “разрешает” конфликт по некоторым позициям случайным законом возбуждения выходных переходов. Однако в подсети ВО могут быть другие конфликтные позиции, в которых возбуждение перехода определяется процессами в УК, например типом внешнего сигнала УК (“ОС” или “СЗ”?). Использование СП для моделирования предполагает разрешение всех конфликтов в том или ином виде. Поэтому при сопряжении подсети ВО с подсетью алгоритма, необходимо выходные переходы каждой конфликтной позиции доопределить инцидентностями из подсети алгоритма, если позиция не объявлена ЖГСЧ. Аналогичное замечание справедливо и для конфликтных позиций подсети алгоритма.

Часто бывает, что максимальное время формирования внешнего сигнала значительно меньше времени реакции абонента (например включение выдержки времени и сама выдержка времени), следовательно описание выдачи этого сигнала можно исключить без большого ущерба точности моделирования. При этом в подсети ВО исключаются переходы, отмечающие это событие и промежуточные позиции, что заметно снижает время моделирования[1].

[1] Использовано в примере — приложение 4.

Важным преимуществом такого подхода к описанию ВО УК является легкость учета различных факторов, влияющих на поток первичных заявок (повторные вызова, нестационарность и т.п.) при помощи элементарного сетевого (графового) описания или программно без привлечения сложных математических моделей.

Обобщим действия по моделированию окружения УК предложенным расширением сети Петри:

1. По SDL-диаграмме алгоритма работы УК или графу подсети алгоритма определить состав входящих и исходящих сигналов из ВО в УК, порядок их возникновения и изменения в форме графа смены сигналов (рис.2.13).
2. По графу смены сигналов образовать первоначальную сеть (рис.2.14):
   2.4 Каждой вершине графа поставить в соответствие позицию сети.
   2.5 Каждому пучку исходящих дуг графа поставить в соответствие переход сети.
   2.6 Каждая полученная позиция связывается прямой инцидентной дугой с переходом, который образован по пучку исходящих из нее дуг.
   2.7 Каждому входящему сигналу при дугах графа поставить в соответствие позицию сети, которая связана обратной инцидентной дугой с подсетью алгоритма УК и прямой — с переходом сети, образованным по пучку включающему эту дугу.
3. Определить наличие в сопрягаемой подсети алгоритма места локальных циклов и образовать в подсети ВО счетчики (рис.2.18).
4. Рассматривая переходы первоначальной сети как макропереходы произвести их определение с использованием ЖГСЧ (рис.2.15—2.17). Позицию, соответствующую вершине "Занятие" графа преобразовать в “автономный” ГСЧ.
5. Для передачи меток, моделирующих потоки заявок для УК образовать соответствующие позиции и дуги в УК (на рис.2.19 показаны штриховой линией).
6. При приемлемости допущения о том, что максимальное время формирования внешнего сигнала значительно меньше времени реакции абонента — минимизировать подсеть.
7. При необходимости факторов влияния на поток первичных заявок (повторные вызова, нестационарность и т.п.) — модифицировать подсеть.
8. Задать тип и параметры законов распределения ГСЧ, или определить на основании имеющейся статистики.

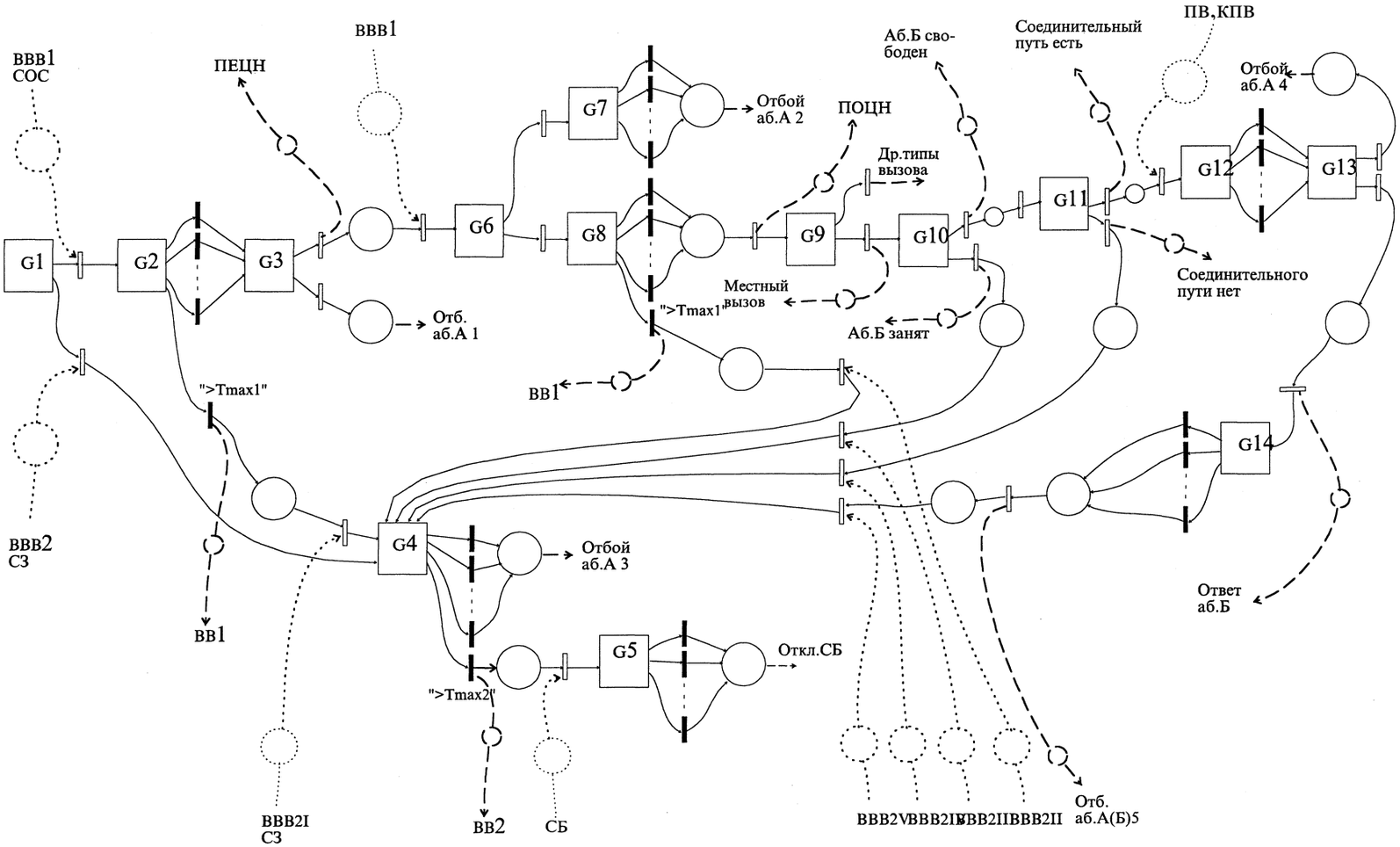


Рис. 2.19 Описание процесса формирования вторичных заявок сетью Петри.

ПЕЦН - первая цифра номера
ПОЦН - последняя цифра номера
ВВ - выдержка времени
ВВВ - включение отсчета ВВ
СЗ, СОС, СБ,ПВ, КПВ - сигналы, занятости, ответа станции, блокировки, посылки вызова и контроля ПВ.

Генераторы вторичных заявок (G) : 1-занятие, 2 - отбой 1+ВВ1+ПЕЦН, 3 - разделение ПЕЦН и отбоя 1, 4 - отбойЗ+ВВ2, 5 - снятие блокировки, 6-ПОЦН+отбой2+ВВ1, 7 - отбой2, 8 - ПОЦН+ ВВI, 9 - тип вызова, 10 - свободность аб.Б 11 - нвличие соед. пути, 12 - ответ аб.Б+отбой аб.А, 13 - разделение ответа Б и отбоя А, 14 - отбой после разговора,

СП
О>

2.3. Сетевое описание структур управляющих устройств узлов коммутации

Отражение конкретного организационно-технического построения УК в сетевой модели предполагает нахождение некоторого множества элементов СП {P,T,F,Mo} и установление правил ее функционирования таких, что существенные количественные характеристики (ресурсы) компонентов УК и их принадлежность получат свои образы в виде абстрактных объектов СП, а динамика изменения этих характеристик в процессе работы УК будет соответствовать динамике перемещения меток в СП с требуемой степенью подобия. Под существенностью ресурсов понимается требование их наличия для выполнения хотя бы одного частного алгоритма рассматриваемого в модели.

Под ресурсами УУ будем подразумевать всякий объект, который может распределяться управляющей программой (операционной системой) между задачами (вычислительными процессами) УУ [62,63,77]. Различают аппаратурные ресурсы:

- производительность ЦПр (процессорное время) по выполнению коммутационных программ,
- используемая емкость основного запоминающего устройства,
- внешние по отношению к УУ устройства, используемые в процессе обслуживания заявок (каналов СЛ, ПАС, ПНН и т.п.);

и программные:

- библиотеки системных и прикладных программ,
- средства программного управления внешними устройствами,
- трансляторы, компоновщики, отладчики и др. вспомогательные программы,
- прочие программы для решения задач УУ.

Будем понимать далее под ресурсами аппаратурные ресурсы, а наличие тех или иных программных ресурсов учитывать в виде увеличения стоимости первых при оценке различных вариантов УУ и их влияния на временные параметры СП.

В общем случае необходимо представление взаимосвязанными сетевыми моделями коммутационной системы, станционных комплектов и УУ УК. Заметим, что структура УК может быть фиксированной либо изменяющейся. В первом случае события в структуре отсутствуют и сетевая модель собственно структуры не содержит переходы (T=0), являясь частью более общей СП.

Блокирующая коммутационная система (КС) непосредственно образует соединительные пути, которые в некотором смысле являются ресурсом, однако ресурсом случайного характера, величина которого зависит от конкретного номера ранее установленного соединения. От конструкции КС (тип, структурные параметры, схема группообразования) зависит вероятность внутренней блокировки, а следовательно и динамика перемещения меток в СП моделирующей УК. Как известно в теории телетрафика разработаны многочисленные аналитические модели которые позволили с достаточной для инженерной практики точностью получить явные зависимости связывающие вероятность блокировки с параметрами потока заявок для подавляющего большинства используемых КС (см. например [17,30,33]). Поэтому в данной работе не ставится задача моделирования КС сетью Петри, а учет влияния КС на перемещение меток в СП предлагается реализовать в виде ЖГСЧ в подсети ВО, причем вероятности появления меток на выходах определяются принятой аналитической моделью для данного типа КС. Это следствие того, что собственно КС обычно является пассивным устройством и не может непосредственно влиять на алгоритм обслуживания вызова. Соответствующая ей позиция причислена к ВО по методике описания в п. 2.2, так как по отношению к УУ, формирующему алгоритм работы УК, заявки КС являются внешними, что соответствует ее отнесению к ОА базовой модели языка SDL, который моделирует внешнее окружение УК (рис.2.20) [62] и является источником сигналов для УА. Сказанное верно и для описания ресурса СЛ с другими УК.

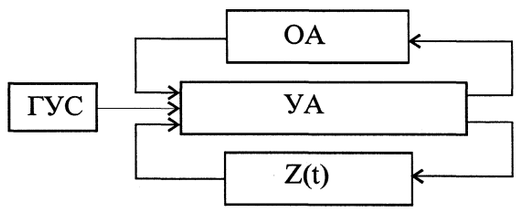


Рис.2.20. Базовая модель SDL:
ГУС - генератор управляющих сигналов,
УА - управляющий автомат,
ОА - операционный автомат,
Z(t) - внутренняя память УК, отображающая предысторию внутренних состояний.

Станционные комплекты (ПАС, ПНН и др.) так же являются ресурсами, используемыми в процессе установления соединения, причем универсальными, так как в отличие от КС, их число не зависит от номера вызываемого абонента и т.д. и определяется лишь числом вызовов, обслуживаемых на данном этапе. Поэтому работу этих комплек-

тов, в соответствии с принятым выше уровнем абстракции, удобно моделировать ресурсной позицией СП, начальная разметка которой соответствует количеству комплектов (линий) доступных рассматриваемой группе источников вызовов, а инцидентности соответствуют частным алгоритмам, которые захватывают и возвращают данный ресурс. Детерминированный характер такой модели (в отличие от ЖГСЧ КС) делает целесообразным отнесение этих позиций к УА, так как занятие свободных комплектов подобно занятию ячеек памяти ОЗУ, отвлечению части производительности ЦПр и т.п. Кроме того сама функция, реализуемая некоторыми комплектами является частью общей функции УУ (например ПНН), такие комплекты имеют свою программу и могут рассматриваться как часть УУ, а не исполнительной системы. Поэтому будем совместно рассматривать УУ и комплекты при построении сетевой модели УУ УК.

Многообразие возможных структур ЭУС УК может быть обобщено в три основных класса, достаточно подробно описанных в литературе [12,30,40,42,77,80,92,94]:

- централизованные,
- иерархические,
- децентрализованные.

В последнем классе, в случае конструктивного объединения отдельных УУ с подчиненными им исполнительными системами, часто выделяют подкласс распределенных ЭУС [89], а в централизованных ЭУС с несколькими процессорами — подкласс многопроцессорных ЭУС. Рассмотренные классы имеют различную структурную организацию и построение сетевой модели по каждому из них естественно будет отличаться.

Для отображения множества существенных характеристик компонентов УК на множество абстрактных объектов СП прежде всего необходимо определить:

- множество компонентов УК обладающих существенными характеристиками;
- характер взаимодействия компонентов между собой, долю и моменты участия каждого компонента в общем алгоритме работы;
- состав существенных характеристик по каждому из компонентов.

Принятие в качестве исходного пункта построения СП алгоритма функционирования УУ УК в виде SDL-диаграммы задает определенный уровень абстракции (степень детальности) сетевой модели в целом. Заметим, что сетевое описание УУ УК не является самостоятельной задачей, а лишь дополняет общую модель, включающую так же опи-

сания ВО и алгоритма функционирования УК. Поэтому к ней предъявляются такие же требования стыковки с базовой подсетью алгоритма, как и к модели ВО (раздел 2.2).

Для определения множества компонент и характера их взаимодействия на указанном уровне абстракции необходим анализ структурной схемы ЭУС для каждой конкретной системы УК.

Централизованные ЭУС

Рассмотрим особенности организации централизованной ЭУС на примере управляющего комплекса “Нева-1”. Структурная схема приведена на рис.2.21

В простейшей комплектации СУВК “Нева” представляет собой ДУК, работающий в синхронном режиме. Все программы приема, обработки информации и выдачи команд выполняются ЦПр. СУВК имеет особенности в организации доступа к ОЗУ и ПЗУ и возможность увеличения производительности при подключении вместо УС периферийного процессора. Помимо станционных комплектов, по данной схеме к ресурсным компонентам следует отнести ЦПр и ОЗУ. Однако это представление не соответствует уровню детализации подсети алгоритма, так как представленные в ней этапы обслуживания вызова захватывают лишь определенную часть ресурса ОсЗУ. Следовательно необходимо указать структурную организацию ОсЗУ — разбиение на массивы и принцип выделения памяти под массивы (фиксированный, плавающий и т.д.). Такие массивы организуются программно, их состав и структура определяется в конкретных случаях по-разному. Однако, практически для всех централизованных УК с коммутацией каналов можно указать общие принципы разбиения ОсЗУ на массивы[1] [77,94]:

1) выделение части ОсЗУ под программное ПрЗУ, которое в свою очередь подразделяется на массивы программ (МП) и массивы данных (МД); справочной информации (МСИ) и констант - МКонст;
2) организация массивов в которых отражается текущее состояние системы: массивы состояния контрольных точек (МСКТ), занятости-свободности (МЗС);
3) организация массивов для реализации процесса выдачи управляющих воздействий, сформированных процессором (процессорами): массив последовательностей перифе-

[1] - названия массивов в технической документации УК могут отличаться, что не влияет на их функциональное назначение.

рийных команд (МППК), иногда выделяется массив сигнальных слов передаваемых по ОКС (МСС);

4) организация массивов для хранения очередей заявок. В зависимости от конкретной реализации могут образовываться либо единый массив заявок на обслуживание (МЗО), либо раздельные массивы для заявок АЛ и СЛ поступающих по ОКС (МЗО ОКС) и от приемников набора номера (МЗО ПНН), либо специализированные массивы заявок: возникающих вызовов (МВВ), повторных вызовов (МПВ) где записываются заявки обнаруженные при повторном сканировании проводимом с целью отсева ложных вызовов, возникающих отбоев МВОТБ, повторных отбоев МПОТБ и т.п.;

5) организация массивов для хранения служебной информации сопровождающей вызов на всех этапах его обслуживания: массив обслуживаемых вызовов (МОВ).

Упрощенный пример типичной несложной структурной организации ОсЗУ с фиксированными полями массивов представлен на рис.2.22.

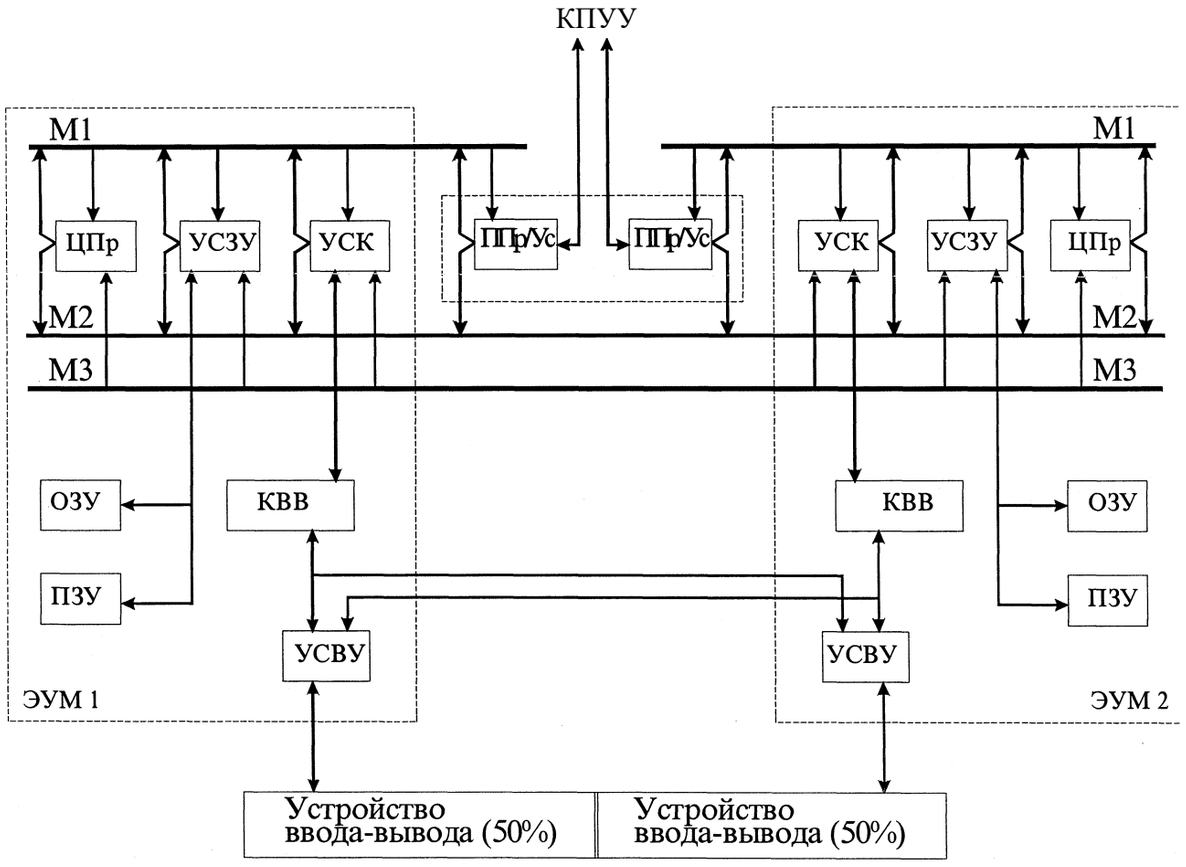


Рис.2.21. Структурная схема управляющего вычислительного комплекса “Нева”

Очевидно не каждый массив из числа приведенных является ресурсом, поскольку такими могут считаться лишь те массивы в которых:

1. Производится не только считывание , но и запись информации связанной с выполняемым процессом.
2. Ячейки памяти массива способны принимать незанятое (свободное от наличия информации) состояние.
3. Ячейки памяти массива не закреплены жестко за устройствами УК.

В этом случае можно считать что часть ресурса массива захватывается процессом при записи в него информации и освобождается по окончании хранения информации (снятии запрета на новую запись). Из рассмотренных, таким требованиям отвечают массивы МППК (исключая МППК сканирования), МОВ, МЗО.

Основное запоминающее устройство

МСКТ	**МП**
МЗО	
МЗС	
МОВ	
МППК	
Счетчики периодических регулярных программ	МКонст
Незанятая область	

ПЗУ

Рис.2.22. Упрощенная структура ОсЗУ

Таким образом определился состав ресурсных компонентов ЭУС: ЦПр, МППК, МОВ, МЗО. Остальные массивы, хотя и имеют важное значение для работы УК, не описываются в сетевой модели УУ УК , так как их количество, структурная организация и объем хранимой информации определяется структурой телефонной периферии, составом программ ОЗУ и т.д., но не самим процессом функционирования, инициированным ВО. Использование этих массивов по своему характеру не является ресурсным, но наряду с такими понятиями как система команд, аппаратурная реализация процессора и т.п. оказывает влияние на производительность ЭУС, а следовательно на такие элементы сетевой модели как временные задержки переходов At и начальная разметка сети M_o.

Каждому ресурсному компоненту ЭУС в СП должна соответствовать позиция, которую в дальнейшем будем называть ресурсной. Отношения инцидентности каждой позиции с переходом сети алгоритма устанавливаются таким образом, что прямые ин-

цидентные дуги связывают позицию со всеми теми переходами подсети алгоритма, которые моделируют частные алгоритмы, захватывающие данный ресурс. Обратные инцидентные дуги должны связать позицию со всеми переходами моделирующими события возвращающие данный ресурс. Пример СП построенной на данных принципах для рассмотренного выше алгоритма представлен на рис.2.24. Точечной линией показана подсеть алгоритма.

Все ресурсные позиции символизируют условия, заключающиеся в готовности некоторого количества ресурса к участию в процессе. Следовательно в момент начала функционирования УК каждый из его ресурсов имеет максимальное значение, что в сетевой модели отражается определенным количеством меток в каждой позиции при начальной разметке. Семантическое содержание меток в каждой позиции определяется вещественной природой ресурсов: число свободных ПНН, число свободных ПАС с данным видом сигнала, число свободных ячеек памяти в массиве, максимальне число одновременно обслуживаемых вызовов. Ясно, что при заданной Мо, независимо от того в каком виде представлены технические характеристики в документации, они должны быть трансформированы в форму, отвечающую правилам функционирования СП. Здесь возникает проблема связанная с тем, что один и тот же ресурс на разных стадиях процесса захватывается в разных количествах, что противоречит неизменному количественному смыслу одной метки. Пример такой ситуации представлен на рис.2.23. Около дуг ресурсной позиции показано относительное количество захватываемого ресурса.

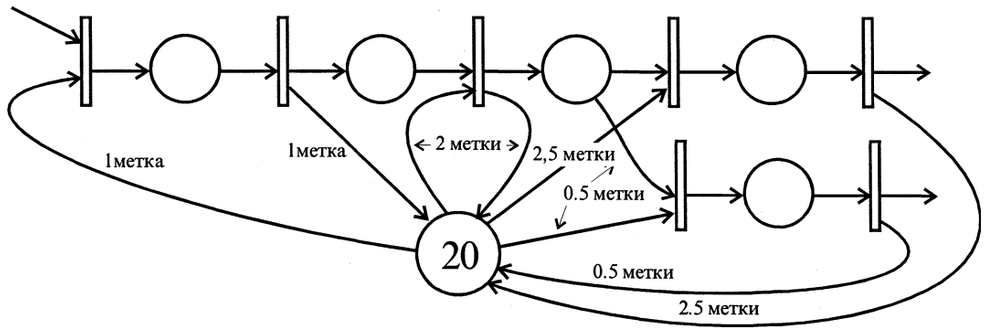


Рис.2.23. К определению кратностей инцидентных дуг

Решением данной проблемы мажет быть “взвешивание” ресурса, когда относительные захваты ресурса каждым переходом представляются отношением натуральных чисел, а начальная разметка ресурсной позиции определяется в таком масштабе, что ресурсная позиция соединяется с переходами пропорционально-кратными дугами (рис.2.25).

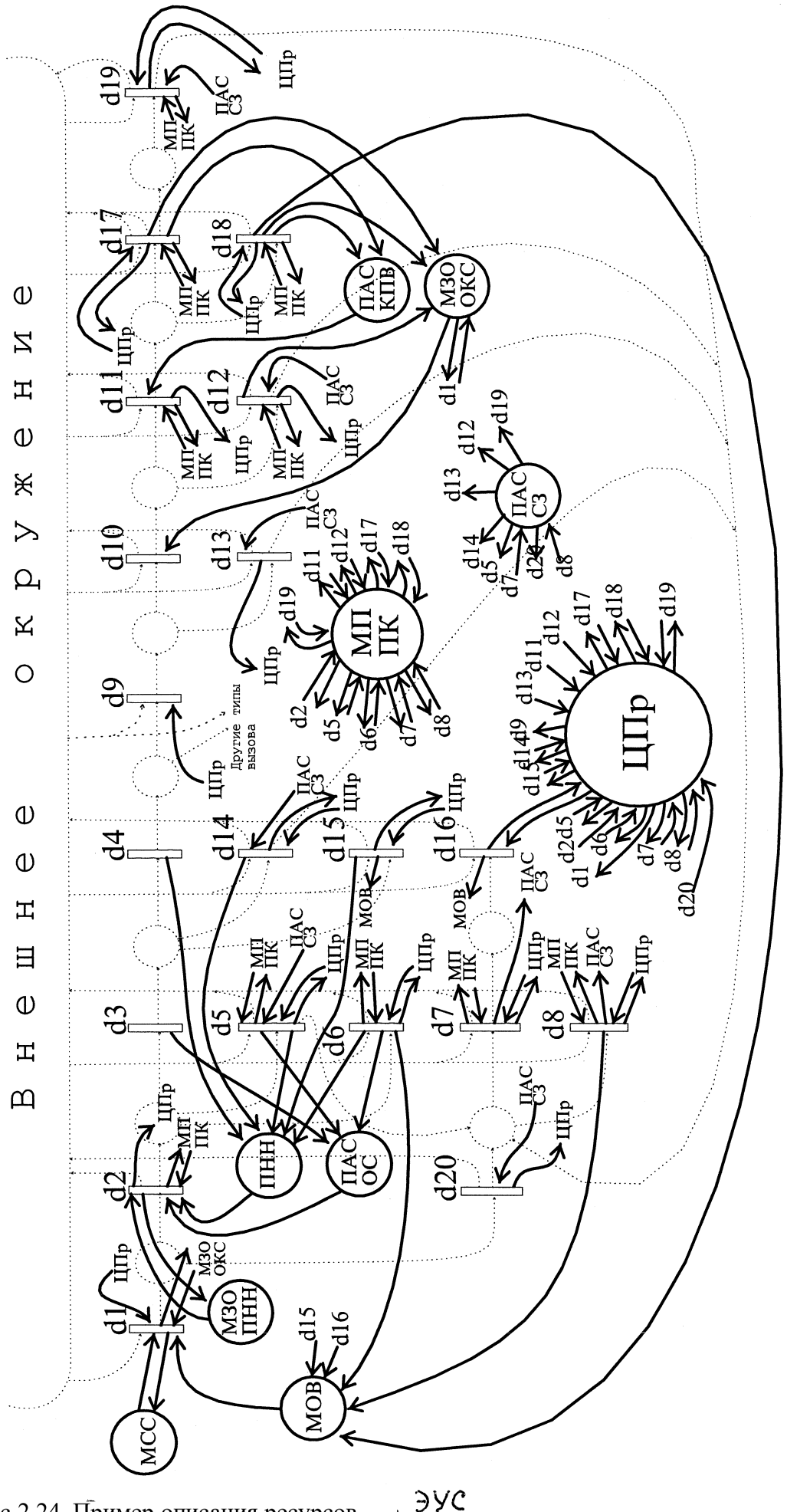


Рис.2.24. Пример описания ресурсов

В некоторых случаях ресурсы в централизованных УК могут иметь неполнодоступной характер включения, и возможны ситуации когда поступивший вызов не может захватить свободный ресурс из-за того что нет доступного ресурса. Можно предложить разные способы разрешения данной проблемы. Первый заключается в том, что поток меток — вызовов от АЛ разделяется на несколько потоков, которые моделируют потоки вызовов одной нагрузочной группы. При этом ресурсные позиции по каждой группе соединяются инцидентными дугами только с теми переходами, которые моделируют обработку вызовов только закрепленной нагрузочной группы (см. пример на рис.2.26)

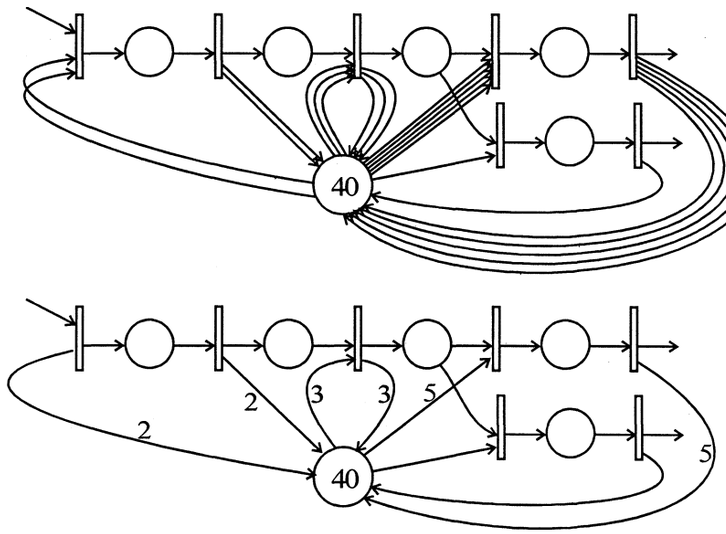


Рис.2.25. К определению кратностей инцидентных дуг

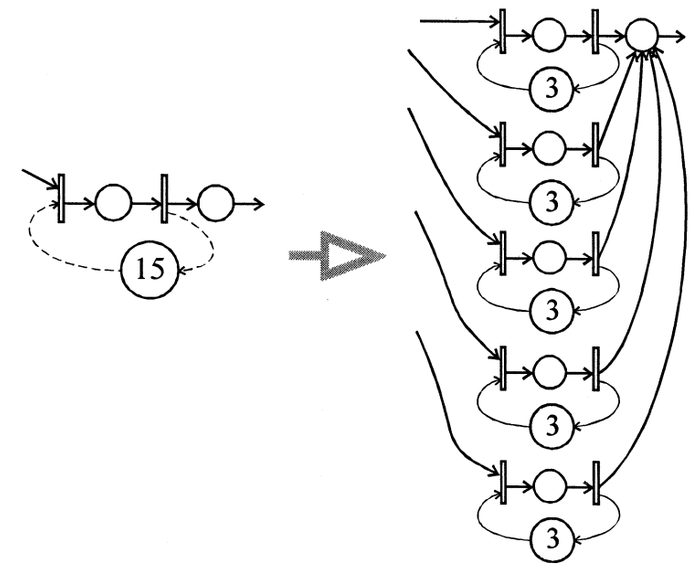


Рис.2.26. Разделение потока заявок неполнодоступного пучка

Очевидно, что при большом количестве нагрузочных групп граф СП значительно увеличивается, что может оказаться недопустимым. Тогда необходимо упрощение модели, например (рис.2.27), установлением вероятностных соотношений между событием “доступный свободный ПНН есть” (вероятность “p”, переход cb) и инверсным ему событием (переход d_3) на основе теории телетрафика. При этом ресурсная позиция заменяется ЖГСМ аналогично коммутационной системе. Такое моделирование самой сети возможно при приемлемости положения о стационарности потока в течении ЧНН.

Способ закрепления ячеек ОЗУ за массивами-ресурсами может быть не фиксированным, а динамическим — “плавающим”, то есть ячейки памяти присваиваются массиву по мере необходимости. Другой возможный способ — “плавающая” граница между двумя соседними массивами, когда емкость ОЗУ, отводимая под каждый массив не является строго фиксированной, а может изменяться в некоторых пределах (рис.2.28)

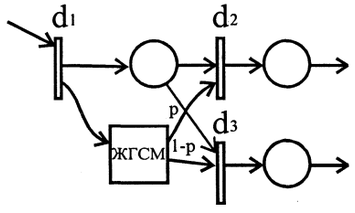


Рис.2.27. Моделирование неполнодоступного пучка ЖГСМ

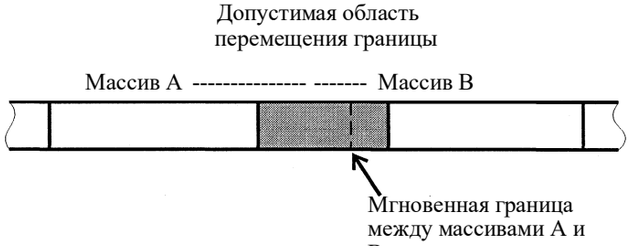


Рис.2.28. Принцип динамического распределения памяти

В первом случае массив можно представить в виде единой ресурсной позиции, ресурс которого захватывается несколькими этапами процесса (рис.2.29)

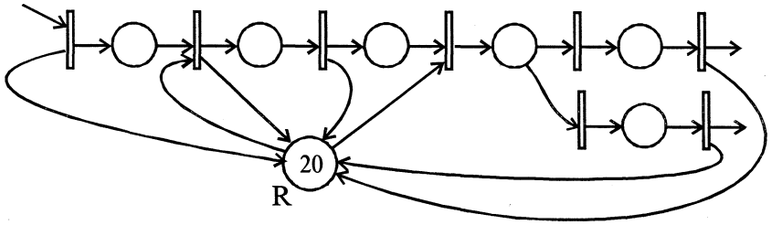


Рис.2.29. Описание ресурса памяти с фиксированным закреплением полей

Во втором случае граничащие массивы можно представить более сложной СП (рис.2.30). Переходы dl и d2 описывают перемещение границы внутри допустимой об-

ласти — передачи ресурса от одного массива другому при исчерпании. Позиции R1 и R2 — ресурсные, а pl и p2 — описывают область перемещения границы. Разметка M(R1,R2) определяет размеры свободных частей массивов, а разметка M(p1, p2) — положение границы между ними и пределы ее перемещения в каждом направлении, Начальная разметка подсети на рис.2.30 говорит о том, что образовано 2 массива с фиксированными полями в 15 ед., допустимой областью перемещения границы в 5+5=10 ед. и средним (5=5) начальным положением границы. Если размеры оперативных единиц в соседних массивах различны можно использовать кратные дуги, аналогично рис.2.23.

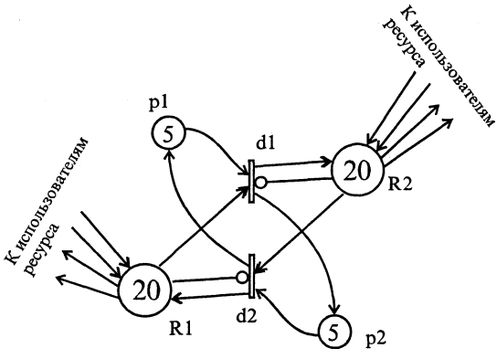


Рис.2.30. Моделирование динамического распределения памяти

Многопроцессорная централизованная ЭУС может строиться по принципу разделения нагрузки и разделения функций, причем возможны варианты фиксированного закрепления частей телефонной периферии за процессорами и незакрепленной процедуры обслуживания. В обоих вариантах логика процесса обслуживания заявок предполагает его распараллеливание — разделение на несколько идентичных параллельных подпроцессов отдельных процессоров. Принципиальное различие между двумя вариантами в моделировании источника заявок: при фиксированном закреплении для каждого процесса источники заявок независимы и могут иметь различные параметры, во втором варианте заявки единого источника направляется в ветви процесса по принятой дисциплине распределения вызовов между процессорами. В случае функциональной организации имеет место поэтапное включение процессоров в обслуживание одного вызова, что соответствует поочередному расположению ресурсных позиций процессоров на сетевой модели алгоритма функционирования УК. Принципы описания других ресурсов такие же как и в ДУК. Примеры, поясняющие описанные принципы моделирования ресурса процессоров многопроцессорной централизованной ЭУС, представлены на рис.2.31.

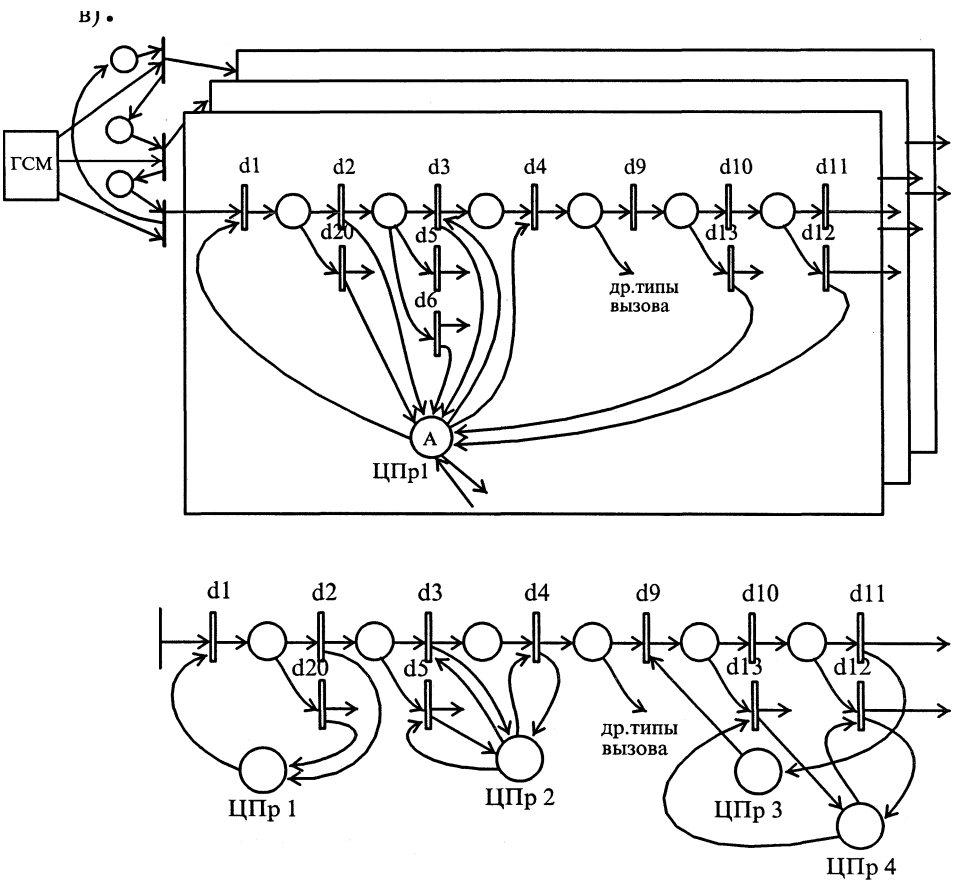


Рис.2.31 Моделирования ресурса процессоров многопроцессорной ЭУС

ГСМ
d1
ожидание СОС
d20
ожидание ННН
d6
d5
ожидание КНН
определение
тапа
др.типы
вызова
определение
свобод-
d15
определение
свобод-
d12
d11
ЯД.р
A

ГСМ
d1
d2
d20
d6
d5
d3
d4
d9
др.типы
вызова
d13
d10
d12
d11
ЦПр1
A

а) Двухмашинный управляющий комплекс;

б) Многопроцессорная ЭУС с разделением нагрузки и закреплением телефонной периферии за ЦПр;

в) Многопроцессорная ЭУС с циклической процедурой распределения вызовов между ЦПр;

г) Многопроцессорная ЭУС с разделением функций.

Заметим, что в многопроцессорных системах в отличие от ДУК режим работы с разнесением фаз обмена и обработки информации во времени не выгоден и не используется, что приводит к возникновению конфликтов при одновременном доступе к одному ОЗУ разных ЦПр. Возникновение такой ситуации носит сложный стохастический характер, так как в каждом случае зависит от конкретного номера набранного абонентом, моментов возникновения потребности доступа к ОЗУ и т.д. В ряде внутренних интерфейсов ЭУС реализованы специальные схемы и методы управления доступом к общим ресурсам, позволяющие эффективно уменьшить потери производительности из-за конфликтов. Принятый выше уровень абстракции общей сетевой модели УК предполагает описание на таком уровне, что данная проблема имеет достаточно локальный характер, подлежит моделированию на более низком (детальном) уровне абстракции, а по результатам моделирования выбирается начальная разметка ресурсов ЦПр с учетом потерь на конфликты. В целом начальная разметка ресурсных позиций определяется исходя из:

- емкости каждого ресурсного компонента (производительность ЦПр, емкость массива, число ПНН и т.д.)
- количества ресурса захватываемого на каждом этапе процесса обслуживания вiзовов и соотношения этих величин для одного ресурса.
- потерь ресурса, не учтенного моделью (регулярные программы, конфликты и т.п.)

Децентрализованные ЭУС.

Достаточно велико многообразие вариантов организации децентрализованных ЭУС, различающихся степенью дробления функций системного интерфейса и др. Обзор существующих децентрализованных УК позволяет сделать вывод о том, что практически их ЭУС используют как функциональный принцип разделения, так и принцип разделения по нагрузке отдельных частей телефонной периферии. Особенности организации децентрализованных ЭУС рассмотрим на примерах УК типа ЭАТС-200 и ITT-1240.

Структурная схема ЭАТС-200 представлена на рис.2.32 [12]. АЛ включается в абонентские модули SUB группами по 64 линии в каждый модуль. Общее емкость оборудования ЭАТС-210 60...3500, а ЭАТС-220 300...64000. В SUB производится аналого-цифровое преобразование, согласование абонентской сигнализации и концентрация нагрузки в отношении 64:30. КС представлена ступенями АИ (SSW) и ГИ (GSW). Удаленные абонентские модули (RSUB) и СЛ включаются в оконечные станционные комплекты (ET), обеспечивающие преобразование линейного кода в станционный, синхронизацию и электрическое согласование сигналов. В выхода ступени АИ, помимо ПЛ к ГИ включаются комплекты конференцсвязи (CNFC) (AONU). В ступени ГИ включены комплекты генератора тональных сигналов (TG) блоки многочастотной сигнализа-4нн(MFSU) и тонального набора НОМера(PBRU). Система управления представлена УУ АМ в каждом блоке SUB блоками УА БАИ (SSU) осуществляющие обработку абонентской сигнализации, управление ступенью АИ и др. — 1 блок SSU на 1SSW, блоками линейной сигнализации (LLSU на 16 линий ИКМ), блоками регистров (RU) способными одновременно обрабатывать 16 вызовов, маркерами (1 блок М на ступень ГИ без учета резерва), 1 блоком системных данных (СМ), 1 блоком статистики (STU), 1 блоком технической эксплуатации (ОМС). Каждый управляющий блок представляет собой микроЭВМ обобщенная структурная схема которой представлена на рис.2.33 [77].

Рассмотренный выше принцип выделения ресурсных компонент для централизованных ЭУС неприменим для децентрализованных ЭУС, так как ресурсные массивы памяти будут распределены по различным управляющим микроЭВМ. В рассматриваемом примере, как и практически во всех известных децентрализованных УК используются стандартные микроЭВМ одного-двух типов, что является одним из преимуществ. Производительность таких машин можно заранее численно охарактеризовать при использовании с той или иной целью, например с привлечением бенчмарковских программ. Вместе с функциональной специализацией это делает целесообразным выбор самих микроЭВМ в качестве ресурсных компонент модели, а задачу моделирования сформулировать в виде: сколько компонент каждого типа и какой производительности необходимо для обеспечения заданного качества обслуживания при некотором варианте внутренней организации УК?

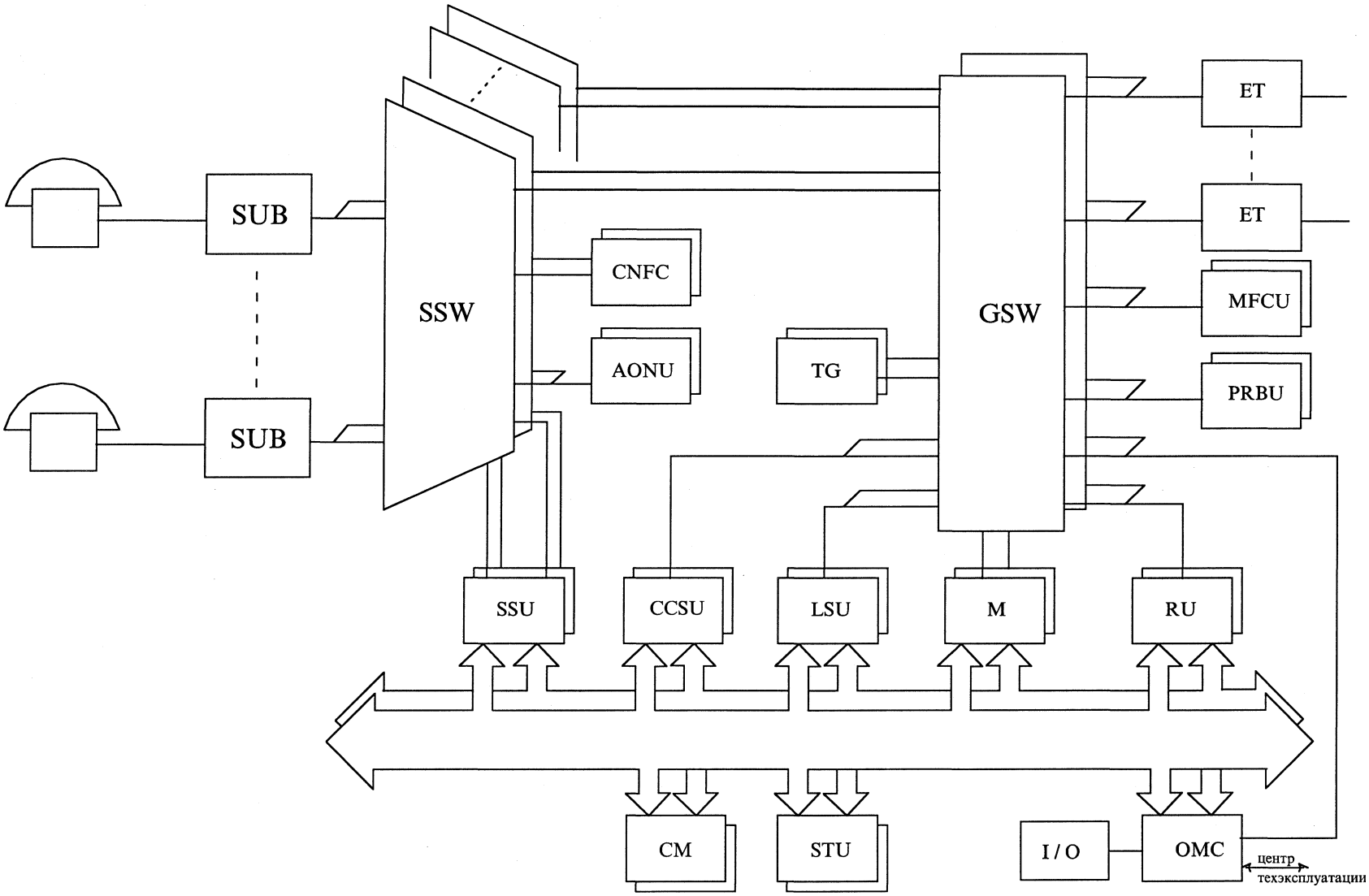


Рис.2.32. Структурная схема УК DX-220

SUB - абонентский модуль, SSW - блок АИ (БАЛ), GSW - блок ГИ (БСЛ), ET - оконечный станционный комплект (КСЛ), CNSF - комплект конференцсвязи, AONU - комплект АОН, TG - генератор тональных сигналов, SSU - блок управления ступенью АИ, CCSU - блок сигнализации по ОКС, LSU - блок сигнализации по ОКС, M - маркер, RU - блок регистров, CM - центральное ЗУ (сист. данные), STU - блок статистики,OMC - блок технической эксплуотации, MFCU - блок много-частотной сигнализации, PRBU - приемники тастатурного набора.

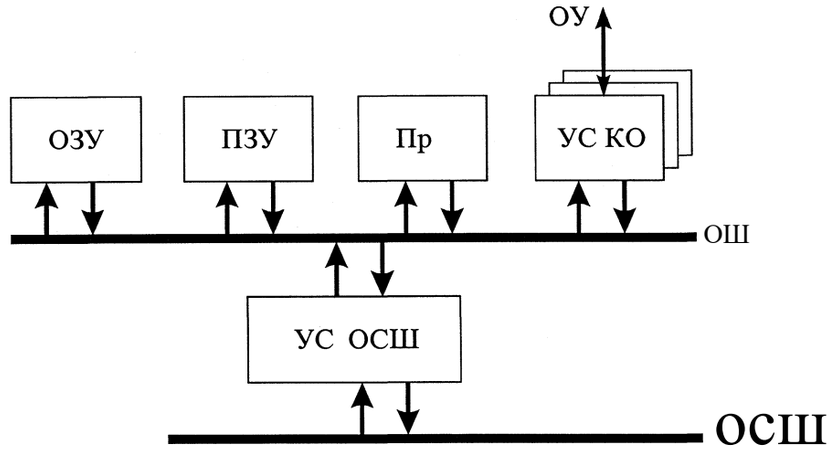


Рис.2.33. МикроЭВМ подсистемы управления DX-200:
Пр - процессор,
УС КО - устройство сопряжения с коммутационным оборудованием и внешними устройствами,
УС ОСШ - устройство сопряжения с ОСШ,
ОШ - общая шина микроЭВМ,
ОСШ- общестанционная шина,
ОУ - объект управления.

В децентрализованной ЭУС, подобно многопроцессорной централизованной ЭУС, поток входящих заявок необходимо представлять в виде параллельных независимых потоков с равными или близкими параметрами, обрабатываемых одним SUB каждый. При закреплении SUB за 64 АЛ при емкости УК 3500№ понадобится изображение более чем 50 ветвей СП. В ряде случаев такая размерность графа СП становится неприемлемо большой с точки зрения объема задачи моделирования — велико время выполнения программы. Для ее уменьшения необходимо принятие каких-то упрощающих предположений и/или соответствующая модификация СП.

В рассматриваемом примере ЭАТС-220 число блоков, работающих по принципу разделения нагрузки относительно невелико (единицы). Поэтому сетевое описание этих компонентов может производиться такими же способами, как и в многопроцессорных централизованных системах. Особенностью представления алгоритма работы УК и его внешнего окружения являются параллельные ветви одного и того же алгоритма, дуги, обеспечивающие обмен метками различных ветвей и раздельные источники заявок для каждой ветви. Заметим что при дублирование СП (ветви) для отражения распределенного характера управления не всегда целесообразно и может быть упрощено использованием ГСМ, а в случае приемлемости гипотезы о равенстве встречных потоков заявок между ветвями — вообще не проводиться.

В практически важном случае, когда групповые потоки заявок идентичны, возможным способом упрощения является пересчет методами теории телетрафика суммарной величины ресурса групповых обслуживающих устройств в единый для всего потока заявок общий ресурс по критерию равенства величины потерь (рис.2.34). При этом группа источников нагрузки заменяется одним, поток заявок которого эквивалентен потоку группы, а групповые ресурсы заменяются одним общим(рис.2.35). Между такими величинами ресурсов можно установить взаимно однозначное соответствие, а показатели качества обслуживания не изменятся[1].

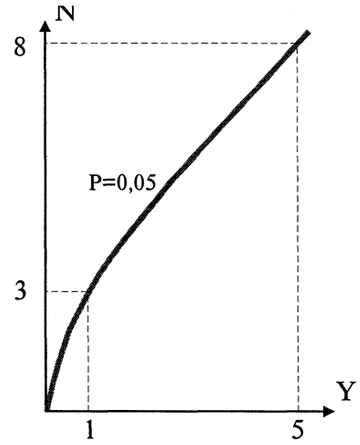


Рис.2.34. Принцип пересчета неполнодоступных ресурсов в полнодоступный

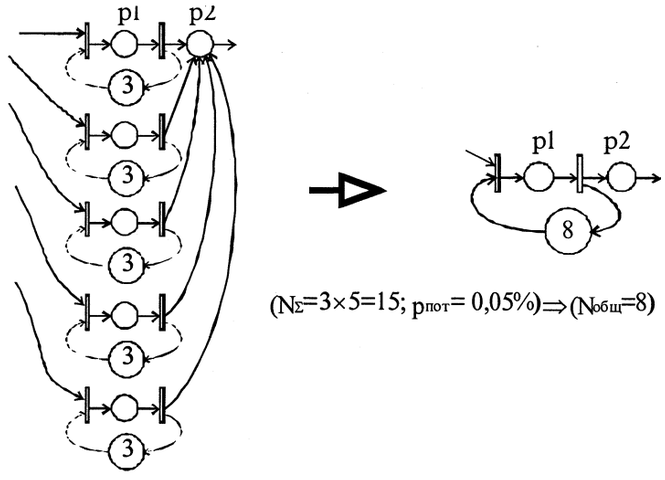
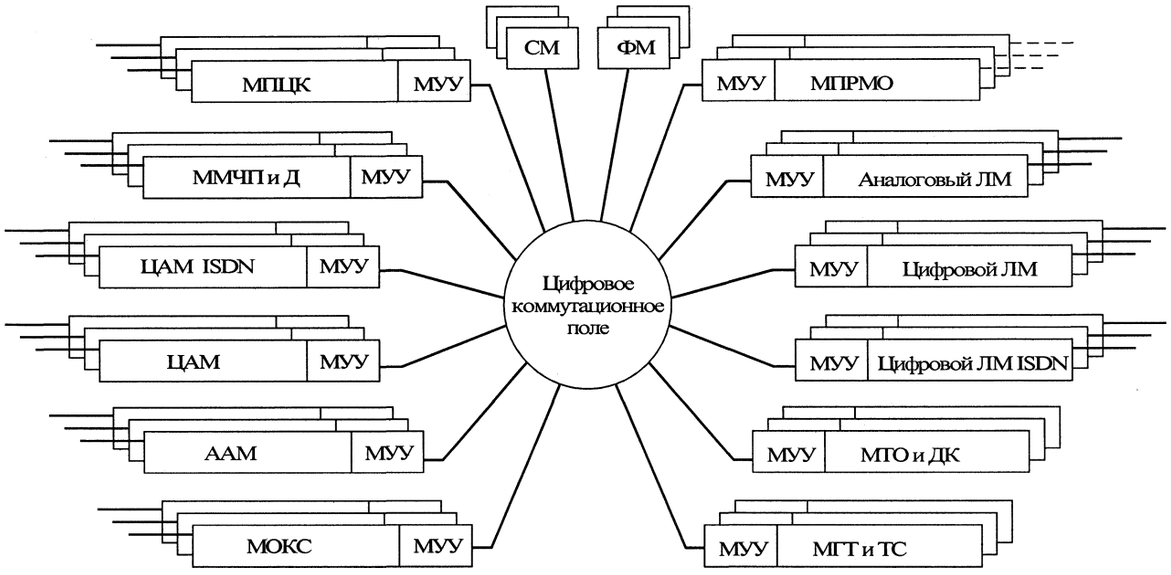


Рис.2.35. Замена неполнодоступного пучка эквивалентным полнодоступным

Структурная схема УК типа ITT-1240 представлена на рис.2.36. УК представляет собой полнодоступное неблокирующее ЦКП, в которое включены функциональные (ФМ), системные (СМ), терминальные (ТМ) и модули приемников и датчиков (МПД),

[1] Модель реального УК ЭАТС-220 с использованием данного принципа рассмотрена в приложении 4.

имеющие собственные УУ. Модуль состоит из микроЭВМ и схем доступа к ЦКП и к устройствам, подключаемым к ТМ (в СМ и ФМ последняя отсутствует). Особенность данного УК в высокой степени децентрализации, распределенном характере управления, возможности работы в составе ISDN, использовании одного ЦКП для передачи, как коммутируемых информационных сигналов, так и служебного трафика между УУ.



СМ - системный модуль,
МПЦК - модуль подключения цифрового концентратора,
ММЧ и ГЩ - модуль многочастотных приемников и датчиков,
ЦАМ ISDN - цифровой абонентский модуль с интеграцией служб ISDN,
ЦАМ - цифровой абонентский модуль,
ААМ - аналоговый абонентский модуль,
МОКС - модуль сигнализации по общему каналу,
МУУ - управляющее устройство модуля,
ФМ - функциональный модуль,
МП РМО - модуль подключения рабочих мест оператора,
ЛМ - линейный модуль (подключения соединительных линий),
JIM ISDN - линейный модуль с интеграцией служб ISDN,
МТО и ДК - модуль технического обслуживания и дистанционного контроля,
МГТ и ТС - модуль генераторов тактовых и тональ-

Рис.2.36. Структура УК ПТ-1240

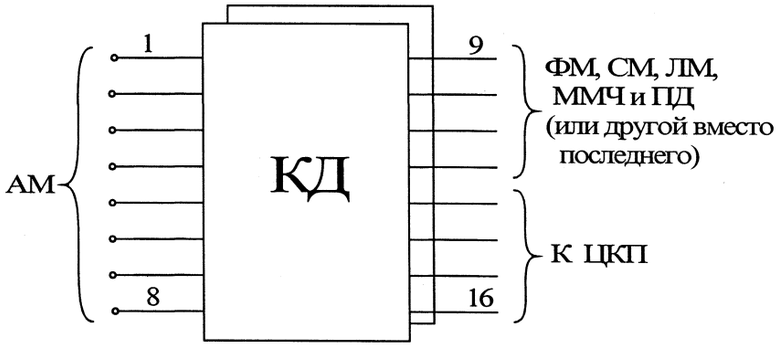


Рис.2.37. Включение модулей в коммутатор доступа

Модули включаются в цифровое коммутационное поле посредством коммутаторов доступа (КД) (рис.2.37)[77]. В один КД, как видно, включается 8 абонентских моду-

лей (АМ) емкостью по 60 АЛ, 4 модуля типов МПД, линейный (ЛМ), СМ, ФМ и линии к остальной части ЦКП. Таким образом функции обработки вызовов распределены пространственно на группы по 480 АЛ, внутри которой пара ФМ-СМ осуществляет управление обработкой вызовов. Таким образом базовой ветвью СП, моделирующей работу данного УК будет ветвь, описывающая обработку вызовов одной пространственной 480-линейной группы. В данном оборудовании в одну базовую группу включено 8 АМ, следовательно имеет место объединение/разъединение 8 индивидуальных потоков меток.

Для сетевого описания рассматриваемого УК можно использовать предыдущий способ, условие применимости которого — идентичность групповых потоков заявок. Как альтернативу ему, можно предложить более сложный способ, при котором данное требование не обязательно. Установим, что результат действия на общий поток заявок всех групповых ресурсов одного типа учитывает одна макропозиция, включаемая в соответствующем месте СП алгоритма. Тогда поток меток можно рассматривать как единый, образованный не 8-ю 60-линейными группами, а одной 480-линейной. Для адекватного моделирования группы компонент, в макропозиции должны иметь место 8 индивидуальных потоков меток, сумма которых должна равняться общему количеству заявок обслуженных данным пучком компонент. Для согласования таких потоков с групповым потоком входящих из внешнего окружения меток, последний должен быть подан в макропозицию. Это можно реализовать, например, путем введения дополнительного “размножающего” перехода с нулевой задержкой, предшествующего каждому переходу подсети алгоритма, с которым макропозиция связана прямой инцидентной дугой. Чтобы отношения инцидентности не противоречили правилам СП необходимо переход подсети алгоритма и “размножающий” переход разделить дополнительной позицией (рис.2.38). Очевидно, что если несколько ресурсных макропозиций связаны с одним переходом подсети алгоритма, достаточно введение одного “размножающего” перехода. Возможная реализация такой макропозиции и ее включение представлены на рис.2.39. ЖГСМ G2 и G2 устанавливают вероятностные законы изъятия и возвращения меток. Каждая входящая в ЖГСМ метка вызывает появление метки на одном из его выходов, что вызывает срабатывание одного из переходов dl.....d8 при наличии меток в соответствующей позиции из ri.....r8. При отсутствии метки в этой позиции, в ЖГСМ

G1 метка передается на девятый выход, где поглощается переходом d9. Позиции p1....p8 обеспечивают “правильность” возвращения меток в ресурсные позиции.

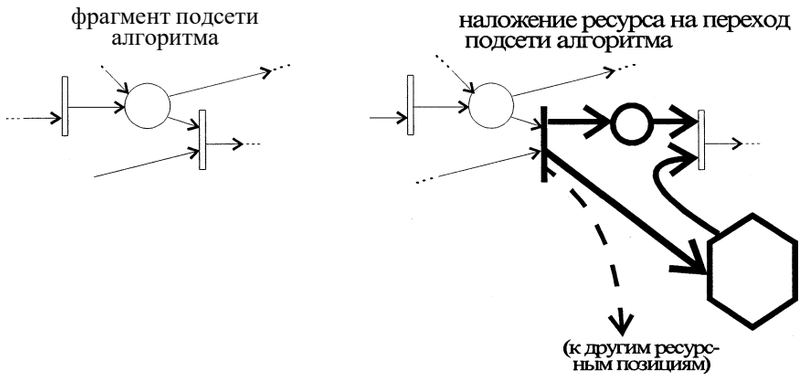


Рис.2.38. Принцип описания неполнодоступного ресурса макропозицией

Базовая ветвь СП, описывающей УУ УК ITT1240 представлена на рис.2.40. Штриховой линией показаны “управляющие” дуги, точечной - подсеть алгоритма. Для меньшей затемненности, вхождение “управляющих” дуг в ресурсные позиции показано однократно. Поскольку рассматривается только местное соединение, состав модулей ограничивается только такими, которые в нем участвуют (АМ, ФМ, СМ, МПД). Состав модулей и их участие в алгоритме местного соединения взяты по [77].

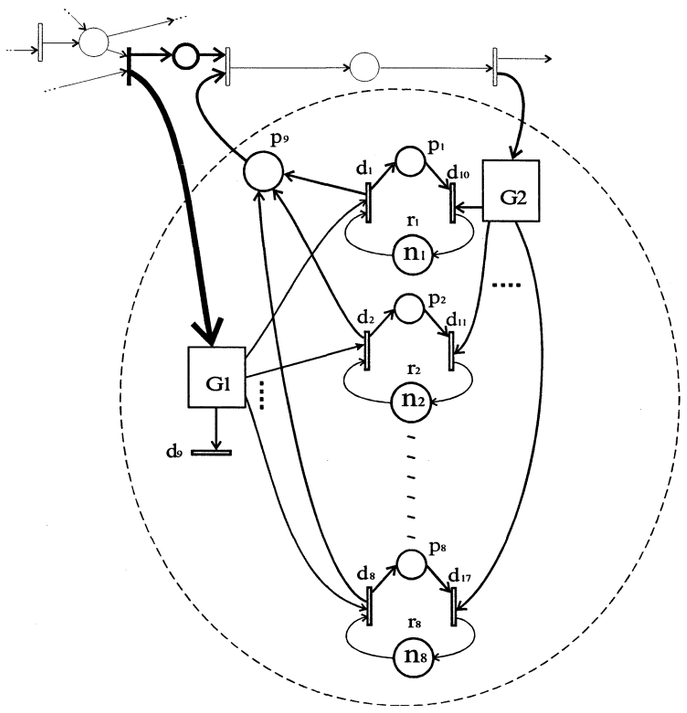


Рис.2.39. Ресурсная макропозиция и включение управляющих и инцидентных дуг

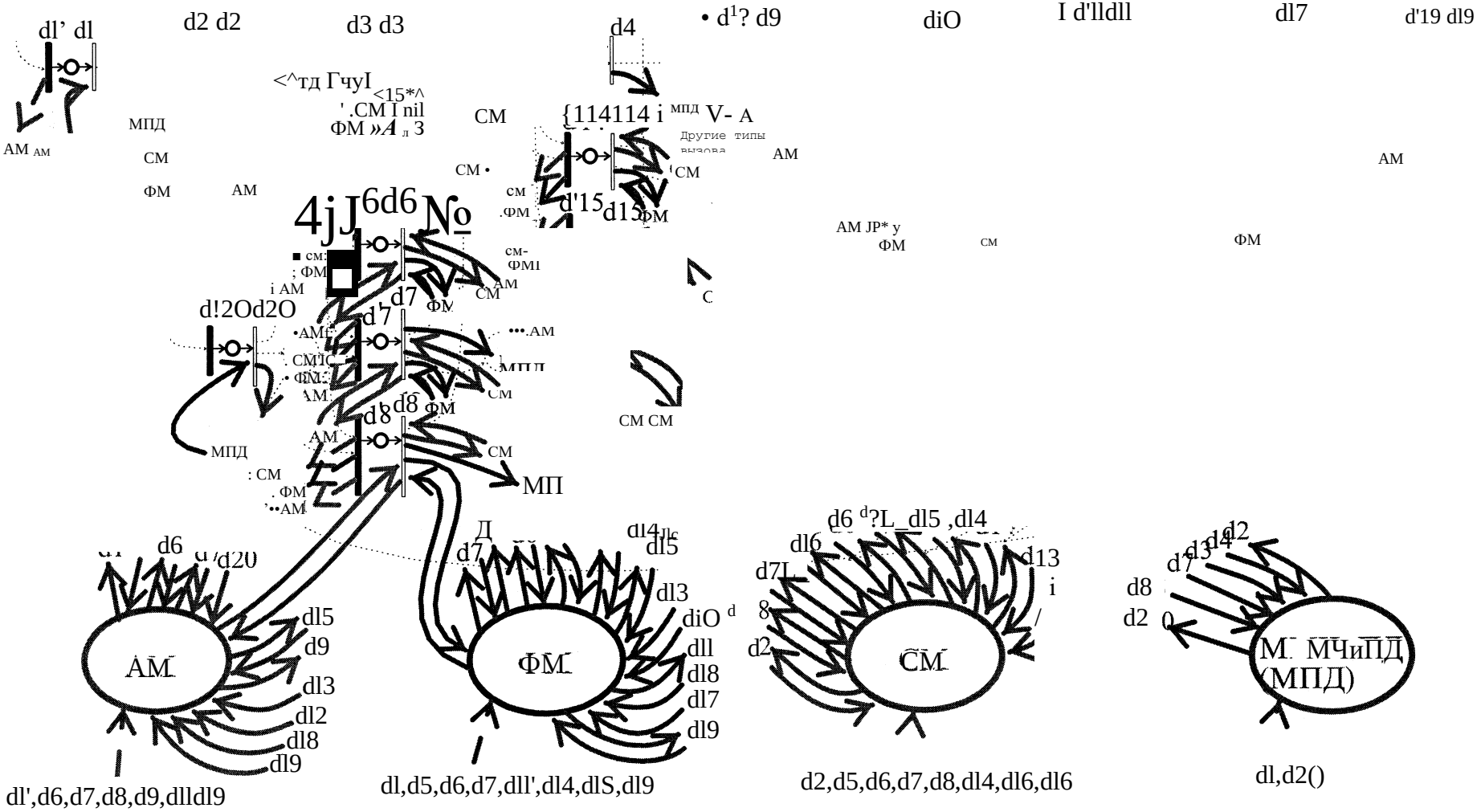


Рис.2.40. Упрщенный пример описания ресурсов УК ITT-1240

Рассмотренное упрощение СП вносит большую графическую избыточность за счет дополнительных элементов — “размножающих” переходов и “управляющих” дуг.

Иерархические ЭУС

Принципы построения ЭУС централизованного и децентрализованного типа совмещены в иерархических ЭУС, в которых имеется ЦУУ, реализующее наиболее общие функции управления и ряд иерархических уровней управления, УУ которых реализуют более частные функции и имеют более массовый характер.

С точки зрения сетевого описания, рассмотренные выше принципы и приемы построения сетевых моделей вполне применимы к иерархическим ЭУС. Особенностью в данном случае является то, что из-за наличия ЦУУ, сеть должна строиться для потока заявок всей телефонной периферии, т.е. на полную емкость УК. Поэтому в сложных случаях необходимо многоступенчатое иерархическое моделирования одним из способов, рассмотренных выше. В целом принципы сетевого описания иерархических УК не имеют каких-либо принципиальных отличий от рассмотренных выше и подробно не рассматриваются.

Анализируя рассмотренные выше примеры можно сделать вывод, что даже в простых случаях граф СП описывающий децентрализованный (иерархический) УУ УК имеет сложную, запутанную конфигурацию из-за большого числа инцидентных дуг. Во многих расширениях СП граф удается значительно упростить введение так называемой управляющей функции которая аналитически задает необходимое условие срабатывания переходов в виде вектора Uk компоненты которого Ujk = {0,1}. Если Ujk=l, то в k-й момент времени разрешается срабатывание перехода dj, при Ujk=0 — переход не срабатывает, даже если во всех входных позициях имеются метки. “Управляющие” дуги в последнем примере носят вспомогательный характер, а именно используются для формирования порядка изъятия и возвращения меток в ресурсные позиции. Если такие дуги заменить управляющим вектором, то логика графического описания процесса обслуживания вызовов не нарушится, а общий вид СП упростится. Тогда при задании СП, кроме описанных выше множеств и функций, должна фигурировать управляющая функция U.

С другой стороны включение всех ресурсных позиций в сетевую модель не всегда обязательно. Отсутствие ресурсной позиции соответствующей какому либо компоненту эквивалентно наличию такой позиции с бесконечной разметкой. Физическая интерпретация такой ситуации — неограниченность данного ресурса, т.е. отсутствие его влияния на пропускную способность УК. Поэтому в тех случаях, когда влиянием того или иного ресурса можно пренебречь соответствующая ресурсная позиция может быть опущена.

Резюмируем рассмотренную методику описания УУ УК в виде следующей последовательности действий:

1 .После построения сетевых моделей внешнего окружения и алгоритма работы УК (разд. 2.1, 2.2), на основе анализа структурной схемы УК определить состав ресурсных компонент, каждому из которых поставить в соответствии ресурсную позицию.

2 .В качестве каркаса СП УУ УК использовать отдельно взятые вершины переходов сетевой модели алгоритма. Проанализировать моменты участия каждого ресурса в алгоритме и отметить их инцидентными дугами (с учетом разделения нагрузки, функций).

3 .Проанализировать закрепление ресурсных позиций за источниками заявок. В случаях неполнодоступного включения ресурсов произвести распараллеливание потоков меток по нагрузочным группам, используя в необходимых случаях обоснованные упрощения (рис.2.34, 2.35, 2.38, 2.39). Скорректировать включение инцидентных дуг в переходы.

4 .Установить, имеется ли динамическое закрепление ресурсных компонент — пары с “плавающей” границей между занимаемыми областями ресурса— и при наличии произвести сетевое описание такого взаимодействия (рис.2.30).

5 .Определить семантическое содержание меток в каждой ресурсной позиции и на основе технических данных, назначить или определить величину имеющегося ресурса с учетом уменьшения ее в каждой позиции на выполнение УК действий не учитываемых сетевой моделью (регулярные программы, потери на конфликты при доступе к общему ОЗУ и т.д.).

6 .Для каждой ресурсной позиции определить соотношения между захватываемыми и возвращаемыми величинами ресурса по каждой инцидентной дуге, представить их отношением взаимно простых целых чисел. По этим числам назначить кратность соот-

ветствующих дуг и выбрать масштаб меток в ресурсных и вспомогательных позициях (рис.2.23,2.25).

7 .При использовании макропозиций ЖГСМ определить закон распределения меток по выходам и установить управляющие дуги по указанному выше правилу.

8 .При использовании управляющей функции определить ее в виде закона распределения номеров переходов, срабатывание которых разрешается при поступлении каждой очередной метки в любую позицию СП алгоритма работы УК, связанную прямой инцидентной дугой с любым из множества выходных переходов данной ресурсной позиции и аналогично для множества входных переходов.

Отметим, что в случае сетевой модели проектируемого УУ УК, когда технические характеристики ресурсных компонент неизвестны и подлежат определению, исходную начальную разметку сети можно назначить так, чтобы количество ресурса в каждой позиции было минимально необходимым для того, чтоб его ограниченность, при заданных параметрах входящих меток, не уменьшала пропускной способности системы в целом. Такую разметку можно определить экспериментально. Для этого необходимо первоначально задаться по каждой ресурсной позиции некоторым конечным количеством ресурса, которое является заведомо большим искомой величины. Затем в течение достаточно большого интервала модельного времени организовать функционирование сети, в процессе которого контролируется текущее значение ресурса в позициях. Разница между минимальным текущим значением и исходным количеством ресурса с учетом доверительного интервала дает искомое исходное значение начальной разметки по каждой позиции. Варьируя данной величиной, в дальнейшем, можно определить влияние ограниченности того или иного ресурса на пропускную способность системы. Более подробно вопросы назначения начальной разметки рассмотрены в п.4.2.1.

2.4 Учет дополнительных факторов влияния

Разработанные выше средства описания УК позволяют учесть и ряд особенностей функционирования УК, не связанных непосредственно с процессом обслуживания вызовов. Таковы, например аварийные ситуации; затраты ресурсов на аппаратурный и программный контроль, тестирование, самотестирование, самообучение, автоматическую реконфигурацию; влияние ошибок ПО, случайных помех (сбоев) и т.п. При необходимости можно предусмотреть описание изменения условий работы УК под влиянием тех или иных факторов. Рассмотрим, например, особенности сетевого описания УК в условиях возникновения сбоев в работе УУ под действием случайных помех и ошибок ПО. По отношению к исправному функционированию УК действие таких дополнительных факторов может проявиться в виде:

- искажений номерной или адресной информации (неверное установление и неустановление соединения);
- потери установленных соединений;
- включение нежелательных соединений (“параллельные абоненты”);
- неправильной выдачи сигналов: ложный сигнал занятости, отсутствие пригласительных, вызывных и контрольных сигналов, сигнал во время разговора;
- уменьшения нагрузочной способности или временного выхода из строя некоторых обслуживающих устройств, в том числе элементов коммутационного поля;
- искажения информации о состоянии (свободности-занятости) обслуживающих устройств.

Нетрудно видеть, что с точки зрения целей моделирования, сформулированных выше, действие таких факторов эквивалентно:

- колебаниям вероятности блокировки КП,
- потере некоторой части вызовов на различных стадиях установления соединений,
- колебаниям величины некоторых ресурсов в процессе работы УК.

Учет воздействий первого типа можно реализовать пересчетом вероятностей ЖГСМ, моделирующего работу КП, второго и третьего типов — путем сетевого описания соответствующих событий.

В первом случае, для определения вероятности блокировки в условиях возможного возникновения неисправности — **p,** необходимо задаться вероятностями блокировки в исправном режиме — **pi** и блокировки в режиме неисправности — **pг,** а также вероятностью возникновения неисправности — **pз.** Введем обозначения: событие А — “при обслуживании заявки нет блокировки”, гипотеза Hi — исправный режим, гипотеза Ш — неисправный режим работы КП. Тогда условные вероятности для события А:

$$X^{\wedge}/Я1) = (1\text{-}p_1),$$
$$^{\wedge}Л/Л_2) = (1\text{-}p_2)$$

Вероятность отсутствия блокировки определится по формуле полной вероятности:

$$X^{л})^{=}{}^{\wedge}X^{я}0'Л^{л/я}0^{=}(1\text{“}Pз)(1\text{-}P1)+Pз(1\text{-}P2) >$$
$$i$$

а искомая вероятность блокировки в условиях возможного возникновения неисправности:

$$p = 1 - p(л) .$$

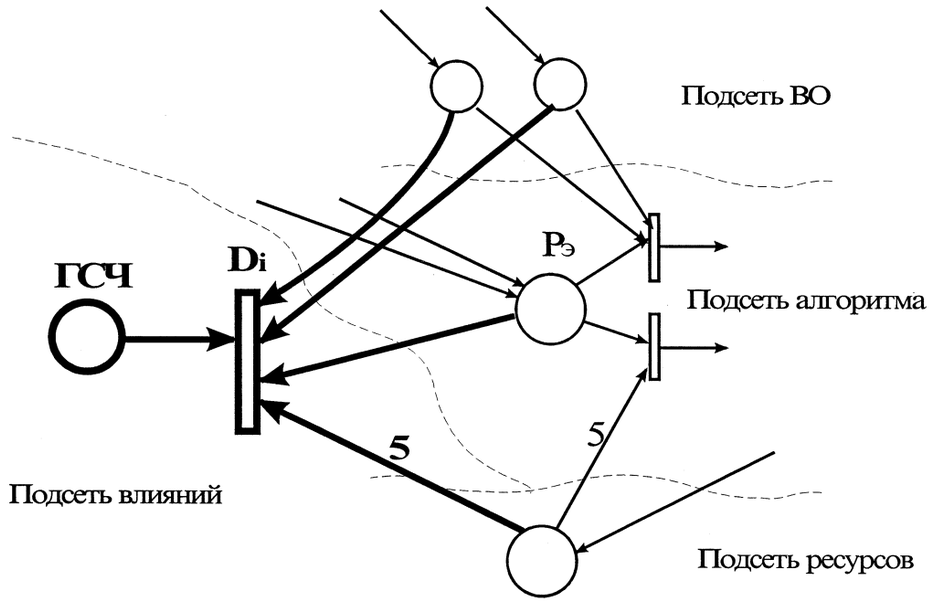


Рис.2.41. Описание потери вызовов

В случае воздействий второго типа в сетевой модели УК необходимо предусмотреть описание аварийного прекращения обслуживания заявки на всех (или интересующих) этапах. Для этого целесообразно создание дополнительной подсети влияний, в которой при каждой позиции $P_э$ подсети алгоритма, соответствующей интересующему

этапу, должен быть образован выходной переход Dj, описывающий указанное событие. Для прекращения дальнейшего формирования вторичных заявок, прямые инцидентности должны связать переход Dj также и со всеми позициями подсети внешнего окружения, которые прямо инцидентны нетерминальным[1] выходным переходам ПОЗИЦИИ Рэ. Для возвращения ресурсов, захваченных алгоритмом к данному этапу, обратные инцидентности должны связать переход Di со всеми соответствующими позициями подсети ресурсов с нужной кратностью. Возникновение аварийных потерь можно моделировать отдельным ГСМ или ЖГСМ, прямые инцидентности которого совпадают с теми, что у позиции $Р_э$, а обратные — к переходу Dj и вновь образованному терминальному переходу. Пример фрагмента СП, поясняющий такой прием показан на рис.2.41.

Для описания колебаний величины ресурсов, связанных с временным выходом из строя отдельных ресурсных компонент, достаточно реализовать отбор меток из соответствующих ресурсных позиций в случайные моменты времени под действием ГСМ (ЖГСМ) (рис.2.42). Извлекаемая и добавляемая часть ресурса отражается кратностью дуг AR, время, на которое отбирается ресурс задается временной задержкой перехода Dj. Как и в предыдущем случае, новые позиции и переходы целесообразно разместить в подсети влияний.

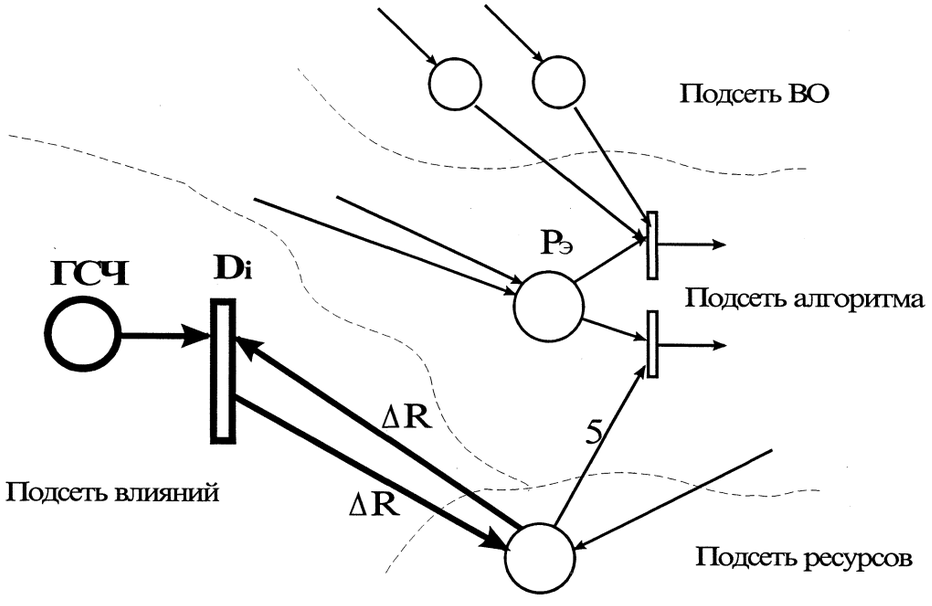


Рис.2.42. Описание временного уменьшения ресурса

[1] Терминальные переходы соответствуют исправному прекращению обслуживания заявки.

ВЫВОДЫ

В соответствии с поставленной задачей, в разделе 2 разработаны принципы функционального описания цифровых систем коммутации. Принятое в разделе 1 решение о составе исходных данных и использовании СП в качестве математической схемы разрабатываемого метода моделирования определило способ формального описания УК и его внешнего окружения. На этапах спецификации-планирования и системного проектирования алгоритм работы УК должен быть представлен в терминах SDL, базовой моделью которого является взаимодействие управляющего и операционного автоматов при обслуживании одиночного вызова. Поэтому предложено разделить общую сетевую модель УК на части, соответствующие базовой модели SDL:

- подсеть алгоритма работы УК — соответствует программе управляющего автомата модели SDL и определяется алгоритмом SDL-диаграммы,
- подсеть внешнего окружения УК — соответствует операционному автомату с источником заявок и определяется статистикой ВО,
- подсеть ресурсов управления УК — соответствует памяти управляющего автомата и определяется структурными решениями этапа системного проектирования и техническими характеристиками выбранной реализации УК.

Каждая подсеть создается отдельно, после чего они объединяются в общую сетевую модель за счет отношений инцидентности с переходами подсети алгоритма. Сетевое Описание указанных составляющих имеет специфические особенности.

1) Описание алгоритма работы УК предложено провести путем трансляции SDL-диаграммы общего алгоритма в СП. Для этого рассмотрены соответствия между понятиями теории конечных автоматов и языка SDL. Установлено, что устойчивым состояниям управляющего автомата соответствует определение SDL СОСТОЯНИЕ, а неустойчивым — РЕШЕНИЕ, множеству входных сигналов — ВХОД , выходных — ВЫХОД, функциям переходов и выходов — ЗАДАЧА, ЗАПОМИНАНИЕ и дуги SDL. Для перехода к СП определены абстрактные события в ней — “переход к n-й стадии обслуживания вызова”, — и условия их наступления — “нахождения вызова на n-й стадии обслуживания”. На основании этих определений установлены формальные соответствия между SDL и СП и их графическая интерпретация.

2) Логику работы операционного автомата ВО предложено отразить подобными соответствиями его состояний, переходов и элементов СП, однако для учета стохастического характера ВО введены специфические позиции ГСМ и ЖГСМ, которые устанавливают случайную функцию возбуждения переходов. Поскольку все недетерминированные переходы являются конфликтующими, ее можно рассматривать как случайную функцию модификации приоритетов, закон распределения которой задается статистикой ВО. Для отражения временных соотношений событий в операционном и управляющем автоматах предложено введение синхронности и временных задержек СП.

3) Для отражения структурной организации УК предложено выделение существенных компонентов ресурсов управления по критерию необходимости для выполнения хотя бы одного частного алгоритма, определяемого переходом подсети алгоритма. Такой подход соответствует уровню абстракции, задаваемому выбранной структурой исходных данных. Поскольку каждый такой компонент накладывает дополнительные условия выполнения частных алгоритмов, в подсети им соответствуют позиции, инцидентные переходам подсети алгоритма. Если структура и величина компонент управления неизменна (статически закреплена), то события в подсети отсутствуют, и она не содержит переходов, иначе — выделяются соответствующие события и производится их сетевое описание. Для отражения логики захвата и возвращения ресурсов и матрицы приоритетов ОС предложено использование приоритетов в СП, а для задания начальной разметки, кратностей дуг — приведение технических характеристик к размерности обслуживаемого вызова. Особенности разных структурных решений по закреплению за УУ обслуживаемой нагрузки и функций нашли отражение в приемах структурного описания. Разделение функций предложено описывать установкой соответствующих отношений инцидентности к переходам подсети алгоритма, а разделение нагрузки — распараллеливанием ветвей СП, а в сложных случаях — приведением величины ресурса к полнодоступному или моделированием ЖГСМ.

4) Учет дополнительных факторов влияния предложено реализовать введением подсети влияний, в которой размещаются ЖГСМ и соответствующие переходы.

Результатом формального описания по разработанным принципам является графическая часть общей модели УК.

РАЗДЕЛ 3
ОЦЕНКА ПРИМЕНИМОСТИ МЕТОДОВ АНАЛИЗА СЕТЕЙ ПЕТРИ К СЕТЕВЫМ МОДЕЛЯМ УЗЛОВ КОММУТАЦИИ

3.1. Методы качественного анализа сетей Петри

Будем разделять понятия качественного и количественного анализа сетей Петри. Под *качественным* анализом будем понимать исследование сети на основные алгоритмические свойства, наиболее массовые из которых ограниченность, безопасность, консервативность, непротиворечивость, устойчивость, живость и потенциальная живость, терминальность. Под *количественным* анализом будем понимать определение некоторых величин, характеризующих динамику перемещения меток в сети на основе исследования топологии, структуры с учетом начальной маркировки, временных задержек и т.д. При этом состав искомых величин определяется задачами практического приложения сети. Каждая такая величина имеет свою интерпретацию как характеристика моделируемого объекта, например емкости массивов памяти, быстродействие процессора, время задержки при передачи информации, среднее время обслуживания вызова и т.п.

Исследование некоторой СП на алгоритмические свойства можно разделить на три этапа:

1) выделение и формальное определение тех свойств сети, которые имеет смысл анализировать,
2) решение вопроса о принципиальной возможности распознавания выделенных свойств, для данной сети или класса сетей,
3) нахождение оптимального способа распознавания тех свойств сети, для которых это возможно.

Состав анализируемых свойств на первом этапе диктуется природой моделируемого объекта (оригинала). Решение о целесообразности исследования сети на некоторое свойство можно принять после выяснения его семантики на данном оригинале.

Рассмотрим каким же образом можно интерпретировать перечисленные выше массовые алгоритмические свойства сетей для конкретной группы оригиналов — УК.

3.1.1 Смысловая интерпретация и разрешимость основных алгоритмических проблем

Свойство ограниченности характеризует емкость условий, образы которых —[1] позиции СП. Несмотря на различный смысл условий, описываемых ими, количественная мера емкости определяется количественной мерой задач, решаемых процессом данной ветви СП. Для УК с коммутацией каналов такой мерой является вызов того или иного типа, и масштабы меток в любой позиции выбираются такими, чтобы одновременное поглощение и генерация их переходами соответствовали некоторому действию производимому с одним вызовом (для одного), находящимся на обслуживании. Примеры:

1. Поступление сигнала ВВI из внешней среды ВС по 1 вызову;
2. Выдан СОС в ВС по 1 вызову;
3. Три вызова находится на стадии обслуживания “Ожидание начала набора номера”;
4. В массиве памяти МППК имеется свободная емкость, достаточная для обслуживания 25 вызовов на этапе установления соединения “подключение ПНН”, или 20 вызовов на этапе “Включение соединения абонент А — абонент Б” или 10 вызовов на этапе “Блокировка абонента по сигналу выдержки времени 2” и т.д.

Количество меток в некоторой позиции говорит о количестве вызовов (заявок), для которых выполняется условие, символизируемое этой позицией. Тогда, неограниченное возрастание числа меток в позиции в процессе работы сети означает выполнение некоторого условия для неограниченно большого количества вызовов одновременно. Поскольку для любого УК число одновременно обслуживаемых вызовов всегда ограничено, данная ситуация может быть корректной лишь в случае терминальной позиции, которая является источником сколь угодно большого числа меток (заявок), или выполняет роль счетчика вызовов, для которых было реализовано действие, описываемое входным переходом этой позиции, с момента начала функционирования сети. Иначе, такую ситуацию следует признать некорректной, явившейся следствием одной из причин:

- ошибок в топологии сетевого описания УК,
-ошибок, просчетов в разрабатываемом алгоритме функционирования УК,

-несоответствия временных задержек переходов параметрам потока вызовов (например УУ УК не “справляется” с потоком заявок при дисциплине обслуживания с ожиданием из-за малого быстродействия);

-несоответствия начальной разметки ресурсных позиций параметрам потока вызовов, т.е. длины очередей заявок неограниченны из-за недостаточности ресурсов.

Можно сделать вывод о целесообразности анализа сети-модели УК на ограниченность.

Свойством безопасности могут обладать СП, в которых нет кратных дуг и начальная разметка позиций не более единицы. Очевидно данный подкласс сетей не может быть отнесен к классу безопасных, и анализ его на безопасность не нужен.

Свойство консервативности предполагает равенство величины начальной и всех доступных разметок степени реализации сети, то есть общее число меток остается постоянным в процессе ее функционирования. Наличие переходов с различной кратностью прямых и обратных инцидентных дуг говорит об однозначной неконсервативности предложенного класса сетей, что исключает необходимость анализа сети в целом на такое свойство. Однако при отдельном рассмотрении подсети алгоритма функционирования УК, или СП частных алгоритмов такое свойство говорит о постоянстве числа обслуживаемых заявок и может приниматься во внимание при некоторых допущениях.

Свойство непротиворечивости (детерминированности) определяется как равенство числа переходов возбуждаемых произвольной разметкой М числу переходов срабатывающих при той же разметке М. Отсутствие такого свойства говорит о стохастичности процессов в сети. Сетевая модель УК предполагает описание внешней среды — источника входных сигналов случайного характера. Кроме того, методика описания УУ УК, которая предложена в разделе 2, допускает использование случайных процессов для упрощения описания ресурсов. Несмотря на это, реакция УК на любую возможную последовательность входных сигналов должна быть однозначной, что соответствует свойству непротиворечивости сетевой модели. Описание случайных процессов при помощи ЖГСМ, которым присвоена случайная функция выборочного возбуждения выходных переходов, позволяет считать их неконфликтными, так как логика работы ЖГСМ гарантирует условие равенства числа возбужденных числу сработавших переходов, и что позволяет непротиворечивой сетью описывать как случайные процессы во внешней среде, так и детерминированные процессы в УК. Невыполнение условия непротиворечиво-

сти в новом смысле говорит об ошибках при построении СП алгоритма, УУ или неверном формировании взаимосвязанных потоков заявок. Таким образом анализ рассматриваемого класса СП на непротиворечивость целесообразен.

Свойство потенциальной живости предполагает достижимость возбуждающих разметок для всех переходов сети из начальной разметки, свойство живости — потенциальную живость всех переходов при любой достижимой разметке. Каждый переход предложенных сетей моделирует некоторое действие, производимое УК или внешним окружением многократно, неограниченное количество раз. Поэтому корректно построенная сеть однозначно должна быть живой. Возможность возникновения тупиковых разметок говорит о том, что существуют такие последовательности действий, в результате которых некоторое действие в УК или во внешнем окружении становится невыполнимым в дальнейшем, что противоречит их физическому смыслу. Следовательно анализ сети на живость позволит выявить наличие некорректных фрагментов в ней.

Свойство терминальности [66] предполагает, что для всех допустимых начальных разметок разомкнутой сети, входящие метки в процессе функционирования покидают сеть или собираются в некоторых терминальных позициях.

Свойством терминальности обладают сети, описывающие завершаемые процессы. В нашем случае свойство удобно сформулировать по-другому: способность незамкнутой сети восстанавливать в процессе функционирования начальную разметку Mo, или принимать одну из множества допустимых $\{M_{Oi}, M_{02}, \ldots, M_{On}\}$ разметок, при прекращении поступления меток в сеть. Нарушение свойства терминальности говорит о нарушении готовности устройств УК к началу его функционирования, или о возможности возникновения ситуаций, когда поступившая заявка остается не обслуженной. Корректно построенная СП, описывающая УК, должна быть терминальной в том смысле, что входами сети являются ГСМ, описывающие поступление первичных заявок, и при прекращении их поступления в процессе функционирования сети, должны быть обслужены все вторичные заявки, и все ресурсы должны возвратиться в ресурсные позиции, разметка вспомогательных позиций должна принять одно из разрешенных значений. Проверка сети на терминальность может выявить ошибки в построении сети или в самом алгоритме работы УК.

Свойство устойчивости [66] предполагает, что для всех допустимых начальных разметок, сеть, после некоторого функционирования, неизбежно принимает начальную разметку. Такое свойство говорит о цикличности описываемых процессов, им могут обладать только замкнутые СП. Очевидно, что для предлагаемых сетевых моделей устойчивость не представляет интереса.

В силу приведенных рассуждений остановимся на целесообразности анализа рассматриваемого класса СП на следующие свойства: ограниченность, непротиворечивость (детерминированность), живость, терминальность. Сети, обладающие такими свойствами назовем *корректными.*

Проблема ограниченности

Разрешимость проблемы ограниченности для обычных СП с конечной начальной разметкой доказана в [34]. Однако в рассматриваемом классе СП введены существенные модификации, а именно:

1. Временные задержки переходов;
2. ЖГСМ и, возможно, другие управляющие функции;
3. ГСМ — источник бесконечного числа меток;
4. Приоритеты (ингибиторные дуги).

Возникает вопрос о разрешимости данной проблемы для модифицированных таким образом сетей. В [34] указаны 2 основных типа неограниченности позиций СП:

1 .“первичная”, при которой существует такая последовательность срабатываний переходов т = т1т2, что разметка Ml, возникающая после начальной разметки МО в результате последовательности ті и разметка М2 возникающая в результате последовательности т сравнимы и находятся в отношении: М2 > М1. В этом случае неограниченными являются позиции рі для которых: М2(р0 > М1(р0, где i=l,2,... . Такая позиция представляет собой внутренний “генератор” меток в сети;

2 . “вторичная”, при которой в СП имеются неограниченные позиции первого типа и кроме того позиции, в которых неограниченно большое но конечное число меток может возникнуть в результате перехода их из “первичной” позиции, когда в ней уже прекратился процесс их накопления.

Одной из используемых модификаций являются специальные терминальные позиции ГСМ, моделирующие потоки первичных заявок, эквивалентные подсети с неог-

раниченной позицией. Поэтому анализ на ограниченность имеет смысл для сети, в которой такие позиции исключены. Но при исключении источника первичных заявок ни один переход, моделирующий действия по обслуживанию вызовов не сработает, то есть сеть мертва, её начальная разметка — тупиковая и анализировать ее бессмысленно. Для выявления позиций обладающих свойством неограниченности, такую сеть можно преобразовать к виду, пригодному для анализа. Если бы сеть была замкнутой, то при конечной разметке Мо позиции, соответствующей ГСМ, функционирование сети было бы обеспечено за счет возвращения "отработанных" меток в эту позицию. С этой целью можно произвести ***инерацию*** (замыкание входа и выхода) сети, например, с помощью следующего алгоритма (рис.3.1):

1 .Выделить, так называемые, *частично-терминальные* переходы подсети алгоритма, у которых нет выходных позиций в данной подсети, или имеющих единственную выходную позицию, выходные переходы которой в другой подсети.

2 .Позиции ГСМ объявить обычными, новыми позициями.

3 .Соединить обратными инцидентными дугами частично-терминальные переходы в каждой ветви подсети алгоритма с соответствующими новыми позициями.

4 .Установить начальную разметку новых позиций равную максимальному количеству вызовов, которые теоретически могут обслуживаться одновременно на всех этапах процесса обслуживания вызовов в соответствующей ветви процесса. Для этого можно проанализировать разметки ресурсных позиций.

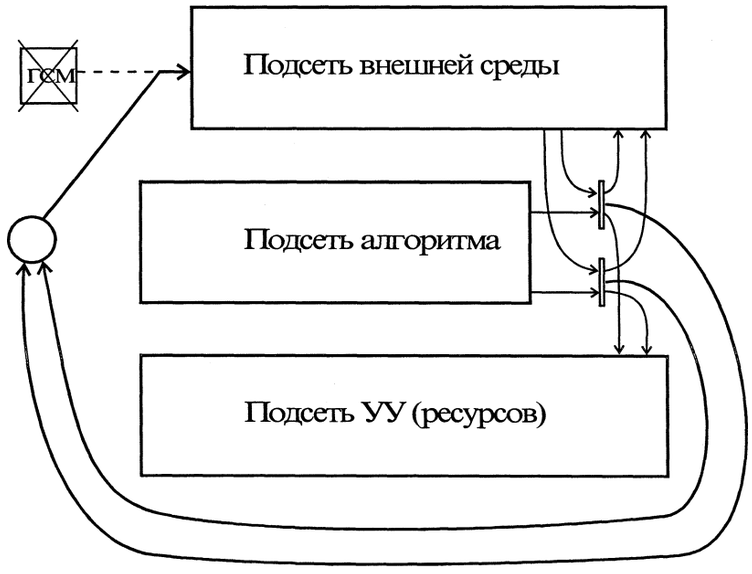


Рис.3.1. Инерация сети

Полученная таким образом сеть моделирует одновременное поступление максимального количества вызовов по системе обслуживания с ожиданием, при равенстве потока освобождений потоку вызовов. В такой модели сделаны значительные отступления от первоначальной, но она позволяет устранить указанную трудность, не дающую анализировать подсети на разрешимость проблемы ограниченности. Такой прием позволяет абстрагироваться от случайного характера поступления вызовов, и сосредоточиться на исследовании топологии самой сетевой модели. Данный прием позволяет так же проверять СП на свойство живости. Можно рассматривать и разомкнутую СП с произвольной (бесконечной) начальной разметкой позиции ГСМ, но тогда имеет смысл анализ только на потенциальную живость и на достижимость тупиковых разметок.

Необходимо выяснить действие временных задержек на пригодность сети к разрешению проблемы ограниченности. Известно, что разрешимость этой проблемы доказана для обычных СП с фиксированной начальной разметкой [34]. Основной идеей доказательства является указание на принципиальную возможность построения покрывающего дерева сети с последующим сопоставлением разметок его вершин. Введение временных задержек переходов приводит к некоторому упорядочению их срабатывания, а значит к уменьшению общего количества вариантов последовательностей срабатывания. Можно предположить, что такое ограничение вариантов последовательностей срабатывания переходов не может породить новых маршрутов в СП и новых ветвей покрывающего дерева сети. Одна и та же СП на может порождать различных графов достижимых разметок, поэтому примем очевидным следующее утверждение.

Аксиома 1. Любая достижимая последовательность срабатываний переходов СП порождает единственную достижимую разметку.

Считаем, что логика работы временной сети такова, что временная задержка связана с пассивным состоянием меток в позициях, как предложено в [62], но временные задержки сопоставлены переходам. Тогда, при срабатывании возбужденного перехода, поглощается число меток по кратности инцидентностей из его входных позиций, для которых истекла временная задержка, выдавших их входных переходов. После выдачи переходом меток, для них начинается отсчет времени задержки, присвоенного этому переходу. Таким образом, временные переходы срабатывают мгновенно, а выданные метки некоторое время, не принимают участие в возбуждении и работе переходов.

Лемма 1. Покрывающее дерево временной сети вкладывается в покрывающее дерево той же сети с исключенными временными задержками.

Доказательство:

Предположим обратное, то есть, что существует 2 сети N={B, *D, Ф, H, Mo}* и N_V={B, *D, Ф, H, Mд, и,* v} с одинаковой топологией *{B, D, Ф, H, Mo}* и отличающиеся тем, что переходам сети N_v присвоены временные задержки т , причем покрывающие деревья TR(N_V) и TR(N) содержат несовпадающие вершины: TR(N_V) TR(N). Это значит, что TR(N_V) содержит хотя бы 1 вершину Моп для которой выполняется одно из двух условий:

1) В TR(N) не существует вершины с разметкой Моп :

$$\bigvee_{Z=1}^{n} M_t : M_m * Mi,$$

где n — число вершин в TR(N).

2) Если ДЛЯ некоторых вершин Мк условие 1) все же не выполняется (Мт = Мк) тогда последовательность вершин на пути из корня TR(N_V) в $M_т$ не совпадает с ни с одной последовательностью вершин из корня TR(N) в Мк:

$$\{Mo\,..\,Mк]\ Ф\ \{MO\,..Mm]\ .$$

Выполнение условия 1) возможно лишь если введение временных задержек приведет к возникновению новых последовательностей срабатывания переходов. В обычных СП логика срабатывания переходов такова, что в произвольном состоянии возможно срабатывание *любых* из возбужденных переходов или несрабатывание всех. Введение временной задержки ставит дополнительные условия для срабатывания переходов, возбужденных по правилам обычных СП, что может лишь исключить срабатывание некоторых из них, но не может вызвать срабатывания не возбужденных переходов.

В соответствии с аксиомой и приведенными рассуждениями выполнение условия 1) возможно лишь при наличии таких переходов в N_v, которых нет в N, что противоречит условию леммы. Выполнение условия 2) требует того, чтобы при некоторой разметке введение временных соотношений приводило к возбуждению и срабатыванию новых, по сравнению с обычной СП переходов, что так же невозможно.

Таким образом верно противоположное утверждение, т.е. условие леммы 1.

Теорема 1. Для того чтобы временная сеть была ограничена, достаточно, чтобы асинхронная СП, полученная из нее исключением временных задержек, была ограничена.
Доказательство прямо следует из леммы 1 и доказательства теорем 2.1 и 2.2 из [34].

Заметим, что присвоение временных задержек позициям удобнее с точки зрения формулировки подобных утверждений, так как нахождение меток в позиции в пассивном состоянии ближе к понятиям обычных СП, чем нахождение метки в переходе. Однако, понятие *захвата меток переходами на время* более удобно с точки зрения осмысления и построения модели и принципиально не влияет на истинность приведенных утверждений. Необходимо лишь изменить формулировку так, чтобы понятие разметки сети распространялось и на метки внутри переходов, например путем “приписывания” их позициям, из которых они поглощены на время временной задержки перехода.

Итак, сформулировано достаточное условие ограниченности временной сети. Поскольку по предлагаемой методике построения сетей определение состава вершин и установление отношений инцидентности (топология) не зависит от временных параметров УК, естественно предположить, что неограниченная асинхронная СП всегда будет соответствовать некорректной модели УК. Хотя это предположение строго не доказуемо, можно считать, что сформулированное условие обеспечивает выявление значительной части ошибок, так как противная ситуация, очевидно, может встретиться редко.

Рассмотрим отношение специальных позиций ЖГСМ к качественному анализу таких сетей. В общем случае, обычные СП могут быть недетерминированными. Правила функционирования их предусматривают возможность срабатывания любых возбужденных переходов или их несрабатывания: из нескольких конфликтующих может сработать любой (любые), или не сработать ни один. Функция ЖГСЧ устанавливает специальное правило выбора одного из конфликтующих возбужденных переходов, которое не противоречит правилам обычных СП и является некоторым частным случаем. Поэтому очевидно, что такие специальные позиции при качественном анализе не вносят каких-либо существенных отличий от обычных, и могут отождествляться с ними.

В отличие от ЖГСМ, — ГСМ вырабатывает метки, а не “передает” их. Такое понятие ближе к элементу с бесконечной разметкой ® [34], или к переходу, который не имеет входных позиций и может срабатывать произвольно. Как видно, с точки зрения качественного анализа СП, ГСМ может отождествляться с некоторым элементом СП, и

также не вносит существенных отличий. Иногда ГСМ удобно представить как входную позицию СП, начальная разметка которой может быть любой, но ограниченной.

Приоритеты в СП можно рассматривать как некоторую управляющую функцию, которая, в отличие от ЖГСМ, детерминирована и имеет не локальный, а общий характер. В произвольном состоянии работа приоритетной СП является лишь одним из возможных вариантов работы более общей обычной СП той же топологии. Поэтому любое свойство, действительное для последней действительно и для приоритетной СП.

Таким образом можно сделать вывод, что для разрешения проблемы ограниченности в предлагаемом классе сетей *достаточно (но не необходимо)* рассматривать более широкую проблему ограниченности обычной СП, топология которой совпадает с исходной, а все расширения (модификации) не принимаются во внимание.

Проблема детерминированности

Методика построения сетевой модели (раздел 2) предусматривает, что каждый конфликт переходов в ней должен быть разрешен либо выбором их приоритетов, либо управляющей функцией ЖГСЧ, либо таким множеством входных позиций, что одновременное появление в них числа меток большего, чем разметка конфликтной позиции, невозможно:

$$УМ\ ^K^M(РрЦ(p_к))]<M(p_кy \qquad (3.1)$$

i

Таким образом разрешение данной проблемы сводится к отысканию конфликтных позиций, и, если она не объявлена ЖГСЧ, — сопоставлению приоритетов конфликтующих переходов, а при их равенстве у двух и более, — проверке условия (3.1). Поскольку методика предусматривает сопряжение подсетей алгоритма и ВО от выходных переходов позиций, конфликты которой разрешены, в одной подсети (р')— к выходным переходам конфликтных позиций в другой подсети (р"), — можно считать, что данная проблема разрешима. Для этого достаточно проследить обратные пути сопрягающих позиций и убедиться, что метки могут попасть в них только из позиции р', и нет “размножения” меток в переходах из множества элементов этих обратных путей.

Проблема живости

Строгое доказательство разрешимости проблемы живости для обычных СП пока неизвестно, хотя гипотеза о ее разрешимости является общепринятой. Разрешимость

проблемы потенциальной живости переходов обычных СП доказана например в [34]. Как и рассуждения о разрешимости проблемы ограниченности, доказательство разрешимости проблемы потенциальной живости основывается на принципиальной возможности построения покрывающего дерева СП и последующем анализе его дуг, — если некоторый переход метит дугу хотя бы один раз, то он потенциально жив. Таким образом содержание леммы 1 устанавливает достаточное условие и для разрешимости данной проблемы в классе временных СП.

Наличие приоритетов в СП может исключить возможность срабатывания некоторых переходов, которые в бесприоритетной СП той же топологии были потенциально живыми. Поэтому подобное условие для приоритетной СП является не достаточным, а необходимым. В [34] доказана неразрешимость данной проблемы для ингибиторных и приоритетных СП. Однако рассматриваемые сети являются всего лишь подклассом со значительными топологическими ограничениями. Конфликтными позициями могут быть:

- позиции, моделирующие стадии обслуживания в подсети алгоритма;
- ресурсные позиции;
- позиции в подсети ВО.

Приоритеты имеет смысл устанавливать только для конфликтующих переходов подсетей алгоритма и вспомогательной вычисляющей подсети (см.п.2.1.1 и п.4.2.2). Конфликтующие переходы подсети ВО доопределяются случайной функцией ЖГСМ или сигналами из УК, и не нуждаются в приоритетах. Выходные переходы позиций подсети алгоритма являются таковыми и для ресурсных позиций. Таким образом можно считать, что элементы подсети ВО не могут изменить живости СП из-за действия приоритетов.

В целом, в сетевой модели приоритеты устанавливаются с целью:

- отметки отсутствия, или недостаточности набора ресурсов для выполнения того или иного этапа установления соединения по некоторому вызову;
- отражения приоритетности предоставления ресурсов для выполнения программ частных алгоритмов в условиях одновременного обслуживания многих заявок, например принятие вызова на обслуживание более приоритетно, чем обработка отбоя.

В первом случае конфликтующей является позиция подсети алгоритма, во втором — ресурсная. Рассмотрим первый случай. Предположим, что некоторая бесприоритетная СП

жива. Тогда при назначении приоритетов, для любой пары разноприоритетных конфликтующих переходов по логике построения данных СП, живость перехода с низшим приоритетом возможна при достижимости одной из входных ресурсных позиций перехода с высшим приоритетом нулевой разметки. Причем такая позиция не должна быть входной для перехода с низшим приоритетом:

(V(d, <7') є *Di* (З *р. (р є* Р) л *(р i Rs}i (F(p, dj = F(p,* J')) л (Pr(d) > Pr(J'))) =>

=} (Зр' є *Pi (F(p', ^ОлР(р', d')* = 0):](Pr(d) > Pr(d')) л

л(р' є/Іу))=>(зМ$_г$(Л9:(Ч(Р') = 0)л(М$_г$(р)^0)), (3.2)

где Rs — множество ресурсных позиций.

Возможность исчерпания ресурсов обязательно должна быть предусмотрена, как некоторое подмножество допустимых начальных разметок, — иначе нет смысла моделировать ресурсные отношения. Следовательно выражение (3.2) можно считать условием того, что введение приоритетов не исключит живых переходов.

Рассмотрим второй случай. Каждый конфликтующий приоритетный переход входит в некоторый цикл ресурсной позиции. Условием срабатывания возбужденного перехода с наименьшим приоритетом является достаточность разметки в ресурсной позиции для поглощения меток всеми переходами, что соответствует достаточности ресурса для обслуживания всех принятых заявок в режиме реального времени. При достаточно большой интенсивности нагрузки для некоторых позиций это условие может и не выполниться. Выявление таких ситуаций и является одной из целей моделирования. Мертвость такого перехода, как и в первом случае говорит не об ошибках модели, а о возможном поведении моделируемого объекта.

Таким образом можно считать, что введение приоритетов при выполнении условия (3.2) не нарушает свойства живости предлагаемых СП.

Наличие ГСМ и ЖГСМ принципиально так же не влияет на разрешимость проблемы живости в силу эквивалентности элементам обычных СП.

В предлагаемом расширении СП, подсети алгоритма и ВО разомкнуты, то есть содержат терминальные позиции. Подсеть ВО к тому же ациклична, а подсеть алгоритма может включать лишь локальные, небольшие циклы. Такие свойства легко объяснимы, так как вызова, проходя различные стадии обслуживания, в конце концов покидают

систему. Для таких СП представляет интерес свойство потенциальной живости, указывающее на принципиальную возможность попадания поступившего вызова на любой этап установления соединения, и вопрос достижимости тупиковых разметок.

Описание структур УУ УК в ресурсной подсети, — напротив, имеет циклический характер, так как захват ресурса вызовом носит временный характер. По окончании обслуживания ресурс должен быть возвращен, что соответствует возврату меток в ресурсную позицию. Таким образом каждая позиция подсети ресурсов, совместно с некоторым множеством путей в подсети алгоритма образует циклическую структуру. Если рассматривать ее обособленно, то ясно, что она должна обладать свойством живости.

Все приведенные рассуждения предполагают, что проблема потенциальной живости обычной СП совпадающей топологии разрешима с привлечением покрывающего дерева СП. Однако такой способ бывает чрезвычайно трудоемок и технически сложен. Преодолением данной проблемы может быть функциональная декомпозиция сложной СП на более простые, топологически ограниченные, живость которых легко доказуема, и анализ сохранения живости при их совместном функционировании (см. п.3.1.2).

Проблема терминалъности

Данную проблему можно рассматривать, как частный случай проблемы достижимости. Ее можно сформулировать так: может ли СП, при заданной начальной разметке Мо, достичь одну из разметок М', причем М'(р') = Мо(р')> $^{r}Д^{e}$ Р' — любая нетерминальная позиция. Как известно, проблемы достижимости и живости взаимно эквивалентны [34]. Поэтому можно считать проблему терминальности частично разрешимой.

3.1.2. Использование топологических ограничений

Предлагаемые сети не могут быть отнесены к какому-либо из известных подклассов СП [2,6], получаемых за счет строгих топологических ограничений, что легко показать на примерах. Но чем больше таких ограничений, тем проще подкласс и легче его математический анализ. Некоторым разрешением данной проблемы может служить прием, состоящий в том, что в общей сети по определенному правилу вычленяется подсеть, которая с одной стороны достаточно полно (в определенном смысле) характеризует некоторые функции моделируемого оригинала, или его часть, а с другой стороны топологически ограничена по известному подклассу СП. Это позволит использовать

имеющиеся методы анализа, применимые по данному подклассу, а также особенности топологии, обусловленные методикой построения таких сетей. Назовем данный прием *функциональной декомпозицией.*

Анализ предложенной сетевой модели позволяет вычленить такие ее фрагменты, которые функционируют по общим правилам СП, и представляют интерес для анализа:

1. Отдельно взятая подсеть алгоритма, инциденции с элементами других подсетей, временные задержки, приоритеты исключены. В такой модели не учитываются процессы во внешней среде, ограниченность ресурсов, временные соотношения.
2. Подсеть внешнего окружения (ВО), взятая совместно с подсетью алгоритма, в которой (Ж)ГСМ заменены обычными позициями, инциденции с элементами других подсетей, временные задержки, приоритеты исключены.
3. Фрагменты сети, состоящие из одной ресурсной позиции и совокупности элементов всех простых циклов (по определению в [6]), включающих эту позицию и любые элементы подсети алгоритма (приоритеты, время, инцидентности с другими элементами исключены). Такая модель описывает характер "перемещения" ресурса, отвлеченно от процессов во внешней среде и в УУ УК, другими словами иллюстрирует, в каких последовательностях этапов обслуживания заявок может брать участие данный ресурс.

Примеры описанных фрагментов, соответствующие сети, полученной в гл.2, представлены на рис.3.2 — 3.5. На рис.3.2 пучки дуг связанных с одной вершиной показаны сливающимися линиями для уменьшения затемненности чертежа.

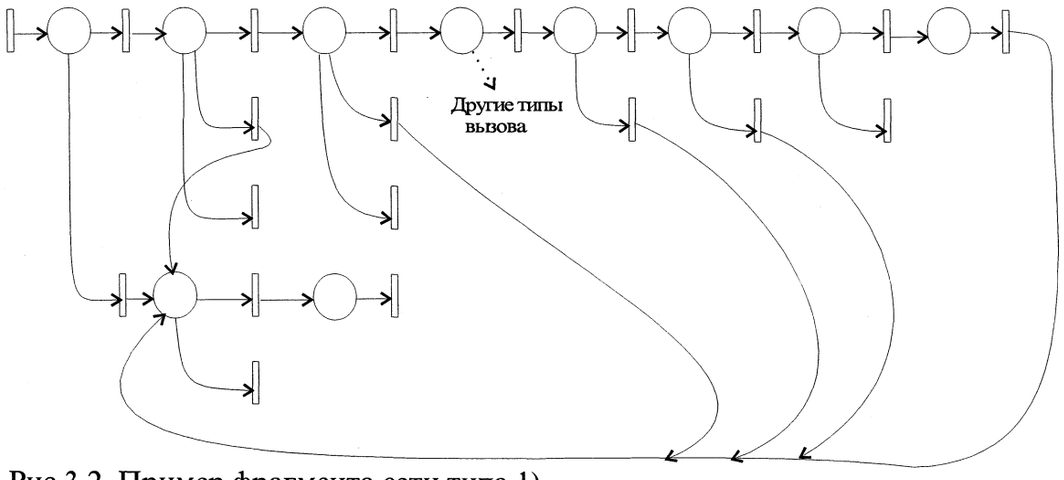


Рис.3.2. Пример фрагмента сети типа 1)

Нетрудно заметить, что фрагмент первого типа (подсеть алгоритма) представляет собой автоматную СП, а если выходные терминальные переходы соединить со входны-

ми через дополнительную позицию — сильносвязную[1] автоматную СП [34] (рис.3.3). Смысл дополнительной позиции — исходное состояние автомата, моделируемого такой сетью, а преобразования указывают на цикличность их смены при обслуживании не единичной заявки, а потока. Свойство сильносвязности говорит о неизбежности возвращения его в исходное состояние после любой возможной последовательности стадий обслуживания. В соответствии с этим, ***допустимая*** начальная разметка позиции, моделирующей исходное состояние, не нулевая, а остальных — нулевая[2]:

$$\forall i \in P': (M(i) = 0) \wedge (M_0 \neq 0), \qquad (3.3)$$

где N — множество натуральных чисел;

i — номер позиции, i є N;

P' — множество позиций СП, исключая исходную.

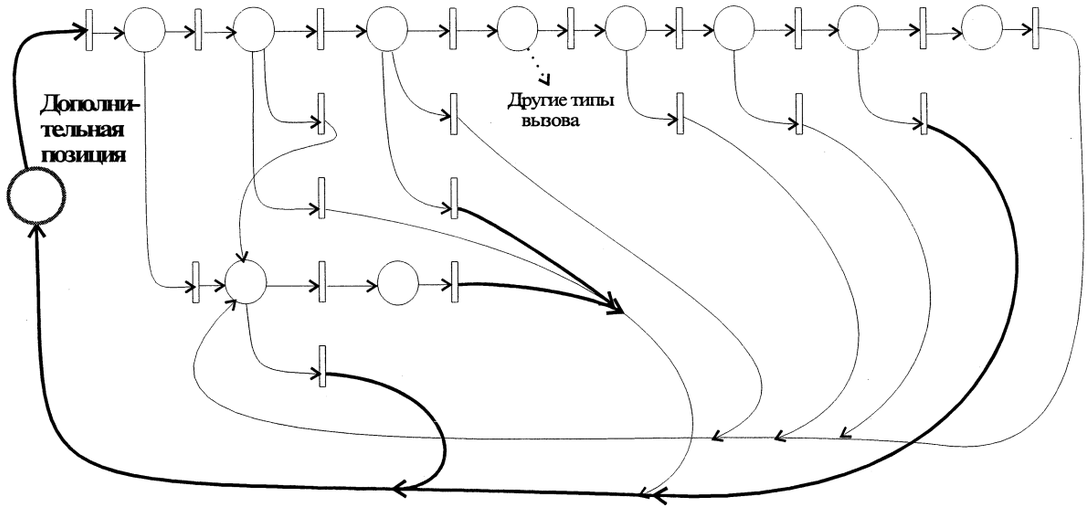


Рис.3.3. Инерация сети на рис.3.2

Ясно, что это прямое следствие того, что методика построения подсети алгоритма предполагает трансляцию в СП SDL-диаграммы, которая в свою очередь строится на основе автоматной модели УК, обслуживающей одиночный вызов. В п.2.1.1, приведены соответствия между терминами SDL и конечных автоматов. Очевидно, что вершины графа разметок такой сети соответствуют состояниям управляющего автомата автоматной модели УК, а переходы сети — символам его алфавита.

---

[1] То есть, что из любой вершины сети существует путь вдоль инцидентных дуг в любую вершину [6]

[2] При описании УК с децентрализованными ЭУС подсеть алгоритма может размножаться, а часть позиций, описывающих распределение вызовов по отдельным УУ — иметь ненулевую Mo (рис.2.31). Такие позиции относятся к подсети УУ УК и во фрагмент 2 не входят.

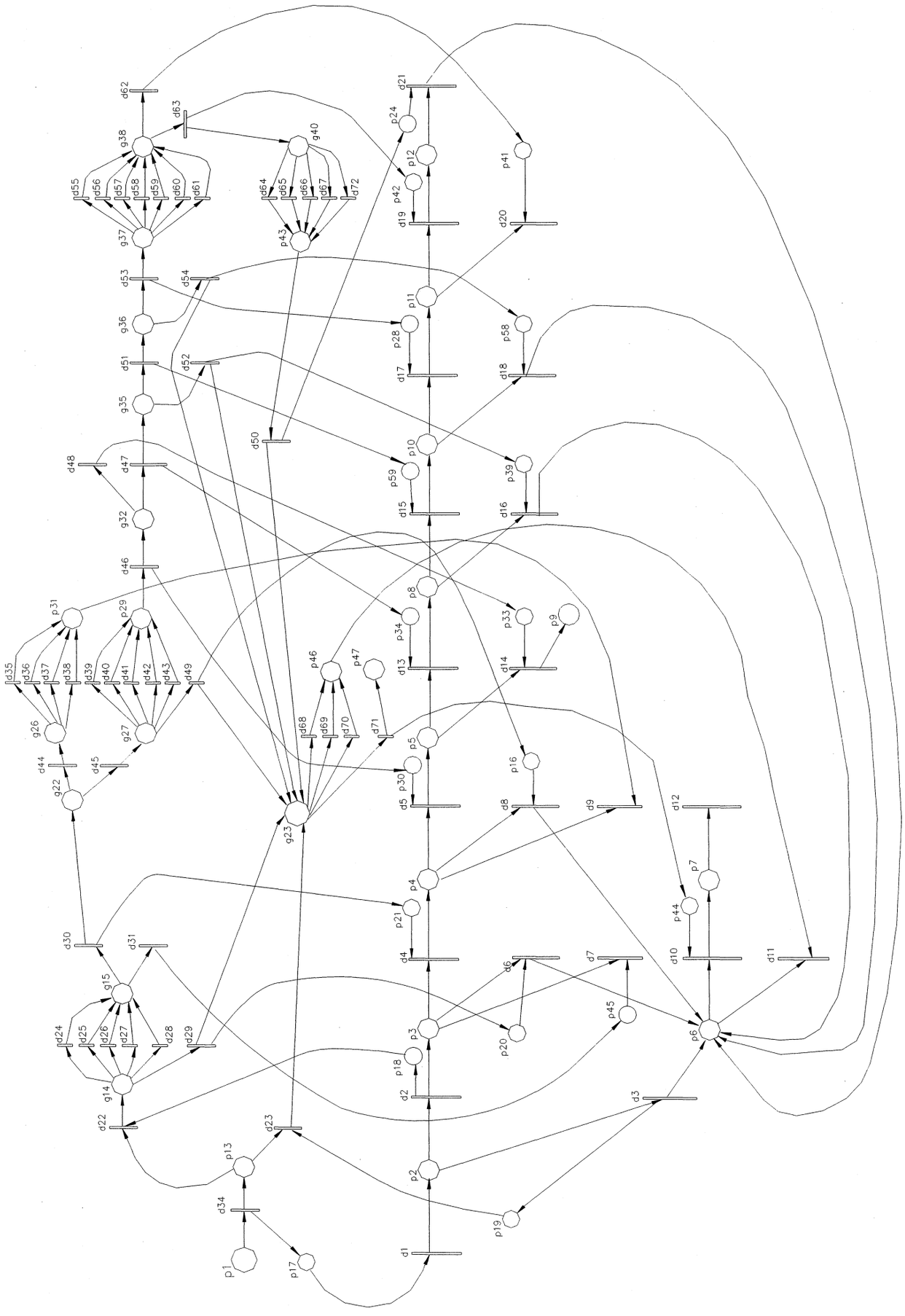


Рис.3.4. Пример фрагмента сети типа 2)

Таким образом можно считать, что для любой сетевой УК, СП, полученная по правилу 1 является автоматной (по построению). Тогда к данной подсети применимы все известные положения об автоматных сетях, а именно:

- граф разметок автоматной СП конечен;
- в классе автоматных СП разрешимы все проблемы анализа свойств СП: достижимости, живости, R-включения и R-эквивалентности, эквивалентности языков и др.;
- автоматная СП жива тогда и только тогда, когда она сильносвязна;
- автоматная СП безопасна, если начальная разметка содержит только одну метку;
- автоматная СП непротиворечива, ограничена и консервативна;
- к автоматным СП применимы методы анализа свободных СП.

Проверка свойств такого фрагмента позволяет выявить часть ошибок и неточностей алгоритма функционирования УК. Принадлежность к классу сильносвязных автоматных СП может служить критерием живости, ограниченности и консервативности сети, и является одним из необходимых условий правильности сетевого описания УК. Определить принадлежность к автоматным сетям по матрицам инцидентности не сложно. Более трудным может оказаться определение сильносвязности. Как альтернативой, можно воспользоваться критерием живости свободных сетей (теорема 4.9 в [6]).

Фрагмент сети второго типа (подсети ВО и алгоритма — рис.3.4), как легко заметить, является ординарной, более того — простой[1] [62] и чистой [62,66] подсетью. Это можно объяснить тем, что подсеть внешнего окружения, как показано в разделе 2.2, моделирует возможные действия абонентов при взаимодействии с автоматной моделью УК при единичном вызове. При этом она выполняет две основные функции:

1) формирование потоков заявок из внешнего окружения,
2) выдачу этих заявок в УК.

Первая функция полностью определяется составом сигналов из УК и правилами взаимодействия с ним. В базовой модели SDL это описывает операционный автомат (ОА), поэтому совокупность внутренних маршрутов в подсети ВО также дает автоматную СП, что (см. пример на рис.3.4). По аналогии с управляющим автоматом (УА), ОА также имеет начальное состояние, однако если первый по окончании обслуживания возвраща-

[1] К классу простых отнесем ординарные СП в которых у каждого перехода имеется не более одной входной конфликтной позиции:Vd) eD:<Vp; щ(p() >1) <11.

ется в него, то операционный, отражая стохастический характер возникновения заявок принимает исходное состояние случайным образом, а не по окончании действий ВО по вызову. В этом случае логично рассматривать не сильносвязную замкнутую модель, а разомкнутую и проверять ее на потенциальную живость и на достижимость тупиковых разметок. Начальное состояние в ОА описывает позиция ГСМ, правила разметки автоматной СП ОА такие же, как и для СП УА — формула (3.3).

Обмен сигналами УК и ВО описывает множество элементов, которые не являются внутренними и не входят в автоматную подсеть ВО. Легко видеть, что правила сопряжения подсетей алгоритма и ВО (п.2.2.) обеспечивают принадлежность фрагмента 2 к классу простых чистых сетей.

Важным моментом является согласованное функционирование подсетей ВО и алгоритма — формирование вторичных заявок в ВО по некоторому вызову должно соответствовать порядку смены стадий его обслуживания. Как было замечено в п.3.1.1, сетевая модель должна быть детерминированной, признаком чего является тот факт, что любой из множества переходов с одной общей позицией в подсети алгоритма обязательно имеет еще хотя бы одну входную "доопределяющую" позицию в подсети ВО или ресурсов. С другой стороны, подобные позиции в ВО “доопределяются” состоянием ресурсов, либо случайной функцией ЖГСМ. Можно заметить что, множество таких позиций ВО, "доопределяющих" одно разветвление, характерно тем, что совокупность всех их обратных простых путей обязательно сходится в некотором ЖГСЧ подсети ВО, и аналогично — обратные пути из ВО сходятся в некоторой позиции подсети алгоритма. Причем множества простых путей до места сходимости не содержат позиций, инцидентных каким-либо переходам вне ЭТИХ путей (см. ПОЗИЦИИ Р4 И g22, Рп и g38, Рiз И Р2 на

рис.3.4). Такая особенность фрагмента 2 не случайна и функционально соответствует ситуации ожидания устройствами УК одного из возможных действий (или длительного бездействия) со стороны объекта ВО, и, аналогично — ожидания одного из возможных сигналов УК. Отсутствие разветвлений вне обратных путей говорит о том, что по каждому вызову, на любой стадии обслуживания обязательно поступает та или иная вторичная заявка, определяющая его дальнейшее обслуживание, и не поступить она не может. То же можно сказать о сигналах УК. Формализуем этот факт, для чего зафиксиру-

Определение 1.

Прямым путем назовем множество элементов w_n = {xi, yi, xг, уг,...} которые упорядочены по направлению инцидентных дуг.:

$$\begin{array}{l} | \ ] \ (x_i \ eP) => (\wedge = \Gamma(x,)); \\ \quad] \ (xj \ \epsilon \Pi) => (\wedge - = Я(XI)). \end{array} \qquad (3\ 4)$$

Определение 2.

Обратным путем назовем множество элементов w_o = {xi, yi, xг, уг, ...} которые упорядочены против направления инцидентных дуг:

$$] \ (^{x}i \ \epsilon P) => (x_г = P(y_i)); \qquad (3\ 5)$$

$$] \ (xj \ eD) => (x_z = Я(YI)),$$

гдеi = 1, 2,... .

Определение 3.

Несколько путей (или прямых w_n, или обратных Wo) ***сходятся*** к элементу z, если все они включают элемент z :

$$\overset{N}{\underset{z=l}{V^{w}ni^{:zew}ni}} \ ' \quad . \quad \overset{м}{\underset{j=\backslash}{\backslash/w_{O}j:zew_{O}j}} \ '$$

где N и M — число сходящихся к z путей.

Определение 4.

Сопрягающей позицией назовем любую позицию, которая одновременно инцидентна переходу подсети алгоритма и переходу подсети ВО.

Обозначим множество таких позиций: Ps = {psi, Ps2, •••}•

Определение 5.

Соответственными позициями (рис.3.5) назовем любую пару конфликтных позиций ***р пр',*** которая удовлетворяет следующим условиям:

1. Множество всех обратных путей от выходных переходов одной из них, которые вклю-

$$Vw_0 \ eW_S(p):\{F(p) \ ew_o)*(p_{Sj} \ ew_o \qquad (3.6)$$

тогда: $p' \ eW_s(p) \wedge > p < - \wedge p'$

2. Соответственная и любая другая позиция, принадлежащая WS, прямо инцидентна только переходам, так же принадлежащим WS :

$(P_K \, ^{c\,w}s \; GO) \Rightarrow (^{F}(Pk)^{c\,w}s \; (P))$

3. Одна из соответственных позиций является единственной входной позицией для своих выходных переходов:

$$Л\,(ж_5(p') = 0) \qquad (Я(Г(P')) = p') \; Л\,(\backslash H(F(P')\backslash = 1) \bullet$$

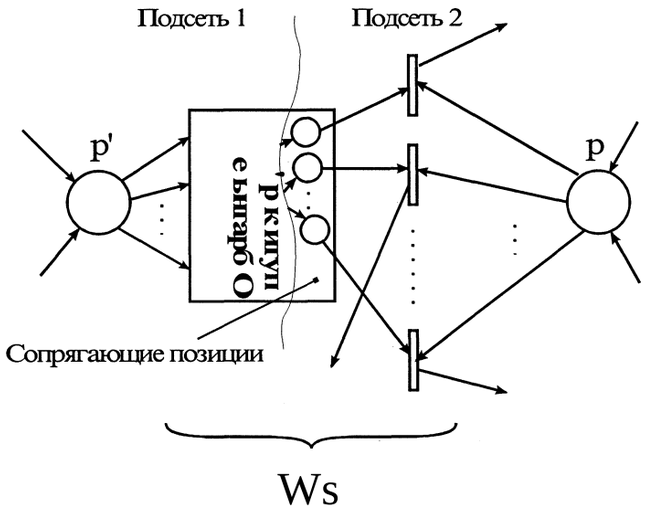


Рис.3.5. К определению соответственных позиций

Данные определения — это особенности фрагмента 2, которые являются теми топологическими ограничениями, которые обеспечивают терминальность общей сетевой модели. Сформулируем еще одно необходимое условие корректности сетевой модели в виде следующей теоремы.

Теорема 2.

Для того, чтобы СП, удовлетворяющая топологическим ограничениям фрагмента2 была жива и терминальна при допустимой начальной разметке, достаточно, чтобы:

1. Все прямые пути от переходов, инцидентных сопрягающей позиции, сходились в соответственные между собой позиции;
2. Любая сопрягающая позиция обязательно входила в множество Ws (3.6) единственной соответственной пары, кроме позиции, сопрягающей входные переходы подсетей;
3. Все выходные переходы одной из соответственных ПОЗИЦИЙ ВХОДИЛИ В Ws-

Доказательство:

При допустимой начальной разметке (3.3) сработать может только входной переход подсети ВО. При этом некоторая сопрягающая позиция получит метку, и сработает входной переход подсети ВО. Тогда по условию 1 теоремы, некоторая пара соот-

ветственных позиций ***{p, p'}*** неизбежно получит по метке. Свойство 1 в определении 5 является необходимым, а свойство 2 — достаточным условием того, чтобы при срабатывании выходного перехода одной из соответственных позиций ***p',*** некоторая сопрягающая позиция в Ws получила бы метку. По построению СП, все конфликты должны быть разрешены, то есть все выходные переходы второй соответственной позиции ***p*** инцидентны некоторой сопрягающей позиции. С учетом условия 2 теоремы и свойства 2 в определении 5, один из выходных переходов ***p,*** так же сможет сработать. По условию 1 уже другая пара соответственных позиций неизбежно получит по метке. Таким образом, появление 1 метки в любой сопрягающей позиции соответствует извлечению по 1 метке из одной соответственной пары и перемещению их в другую пару, причем каждой метке в одной позиции пары соответствует одна метка во второй.

Правила сопряжения живых автоматных подсетей алгоритма и ВО таковы, что инцидентности устанавливаются только от сопрягающих позиций к выходным переходам некоторых конфликтных позиций. Тогда полученная СП мертва только в том случае, если мертв выходной переход сопрягающей позиции. Но любая из них может получить метку, так как жив ее входной переход. Тогда и выходной ее переход жив, поскольку появлению метки в сопрягающей позиции соответствует метка в одной ***(p*** 9, а извлечению — во второй ***(p)*** соответственной позиции. Таким образом вся СП жива.

Число срабатываний входных переходов конечно и равно разметке головной позиции M(po). Тогда и число меток, полученных первыми соответственными позициями, число срабатываний их входных переходов так же конечно и равно M(po). Тогда и в любой другой соответственной паре число меток равно и конечно.

Свойство терминальности требует, чтобы при отсутствии входящих меток и достаточно большом числе тактов работы СП, все метки покидали ее или собирались в терминальных позициях. В живой, ординарной, чистой СП с начальной разметкой (3.3) нетерминальность может быть только следствием того, что достижима такая разметка, при которой некоторые переходы, имеющие больше одной входной позиции не могут сработать, причем хотя бы одна из входных позиций имеет ненулевую разметку. В рассматриваемом классе СП такие переходы в качестве входных могут иметь только одну из соответственных и сопрягающую позиции. При равном числе меток, полученных соответственными позициями, в силу приведенных рассуждений, их разметка в конце кон-

цов станет нулевой. Остальные позиции, кроме конечных, являются единственными входными позициями своих выходных переходов и так же придут к нулевой разметке. Таким образом СП терминальна. □

Теперь можно сделать некоторые выводы по анализу общей сетевой модели на основные алгоритмические свойства:

- часть фрагмента типа 2 без позиций, инцидентных переходам подсети УК (без внешних путей), можно анализировать как автоматную СП, аналогично фрагменту типа 1 (вместо дополнительной позиции — ГСЧ);

- фрагмент типа 2 должен удовлетворять условию принадлежности к классу простых чистых сетей;

- фрагмент типа 2 должен удовлетворять условию теоремы 2.

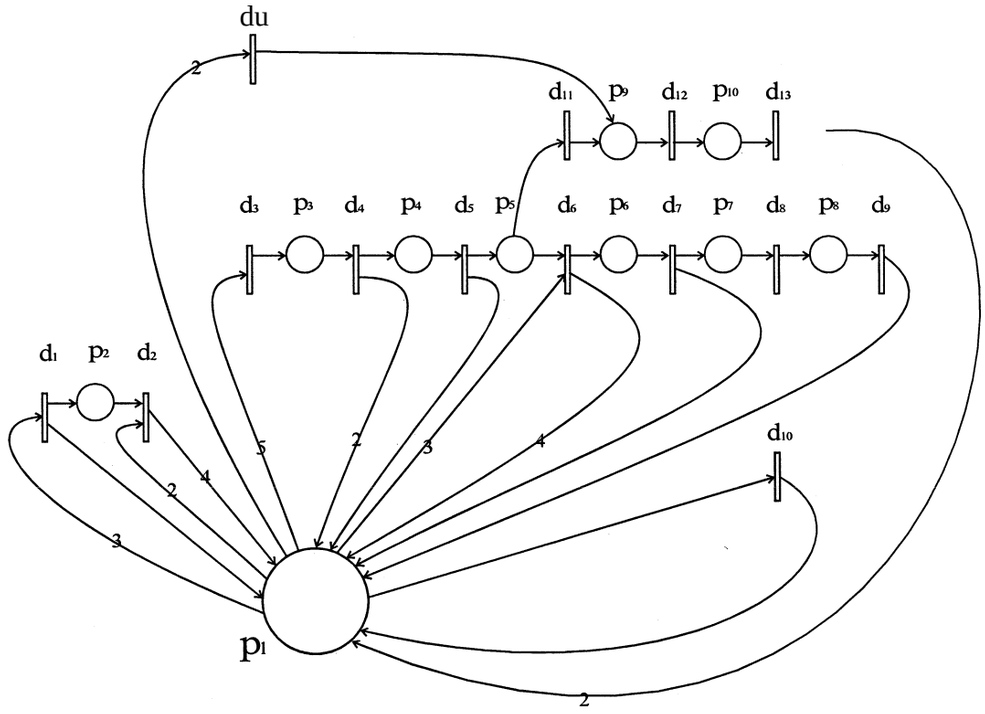


Рис.3.6. Пример фрагмента сети типа 3)

На рис.3.6 показан пример фрагмента третьего типа, в котором рассматривается множество возможных последовательностей стадий обслуживания заявок, использующих ресурс pi. По определению, очевидно, он представляет собой неординарную СП циклической структуры. Семантически это означает, что ресурс циклически захватывается и возвращается в произвольных количествах. Одной из важных проблем, возникающих при отладке сетевой модели, является корректность указания инцидентностей ресурсных позиций и назначение их кратностей. Каждая конкретная реализация УК оп-

ределяет по-своему эти элементы сети, устанавливая, на какой стадии, сколько ресурса данного типа захватывается и возвращается.[1] Неточность здесь может привести к мертвости некоторых переходов или к нарушению циклического характера перемещения меток, что в свою очередь может повлечь убывание до нуля или бесконечное нарастание разметки ресурсной позиции, а значит неправильное функционирование сети и ошибочное определение показателей качества обслуживания вызовов.

Рассмотрим некоторые из необходимых условий правильности назначения кратностей инцидентных дуг. Будем считать, что ограничений на кратности нет, начальная разметка для всех позиций, исключая ресурсную — нулевая, а для ресурсной — ненулевая. Введем несколько понятий.

Определение 6.

Максимальным циклом ресурсной позиции назовем простой цикл, множество всех элементов которого не является подмножеством какого-либо другого цикла.

На рис.3.6 имеется пять максимальных циклов:

Ci^m = (pi, di, p2, d2, pi); $Cг^т$ = (pi, ds, pз, cЦ, p4, ds, ps, du, pg, di2, pio, di_3, pi);

C_3^m = (pi, d_3, p_3, d4, p4, d_5, p_5, d_6, pe, d_7, p_7, d_8, p_8, d_9, pi); C_4^m = (pi, dio, pi);

$C\$^m$ = (pi, di4, P9, di2, Pio, di_3, pi)

Определение 7.

Максимальным классом циклов ресурсной позиции назовем множество, состоящее из максимального цикла и всех циклов, являющихся его подмножествами.

Примеры максимального класса на рис.3.6:

$M^т$ = {СЛ ($_{P1}$, di, pi), ($_{P1}$, d_2, $_{P1}$)}; $X_4^т$ = {$C_4^т$}.

Определение 8.

Началом и ***концом*** цикла называются соответственно первый и последний переходы, следующие после и перед ресурсной позицией.

Определение 9.

Длиной цикла назовем число его переходов.

В цикле все переходы упорядочены отношениями инцидентности и следовательно могут быть пронумерованы от начала к концу натуральными числами, например, в $C_3^т$:

d_3 —>1, cЦ —>2, d_3 —>3, dg—>4, d_3 —> 5,d_8 —>6, dg —>7.

---

[1] Методика определения разметки ресурсных позиций и кратности инциденций рассмотрена в п.4.2.1.

Определение 10.

Номером перехода в цикле будем называть его порядковый номер в последовательности переходов, упорядоченной по отношениям инцидентности, считая, что начало цикла имеет №1.

Номером позиции в цикле будем называть ее порядковый номер в последовательности позиций, упорядоченной по отношениям инцидентности, считая, что позиция, обратно инцидентная началу цикла, имеет №1.

Функциональный смысл цикла в данной сети — последовательность стадий обслуживания, в которой часть ресурса может быть непрерывно захвачена, а максимального цикла — последовательность всех стадий обслуживания, которые могут непрерывно захватить ресурс. В этих утверждениях, верных для общего случая захвата (возврата) ресурса, однако, не учтены кратности инциденций, которые фактически могут быть назначены так, что циркуляция ресурса сможет протекать не по всему максимальному классу, а только по его подмножеству (см. пример на рис.3.7) Такая ситуация соответствует тому, что часть переходов мертвы (на рис.3.7 сіз и сЦ) и могут быть исключены из рассмотрения, а оставшиеся части максимального цикла как раз будут соответствовать указанному функциональному смыслу (для максимального цикла).

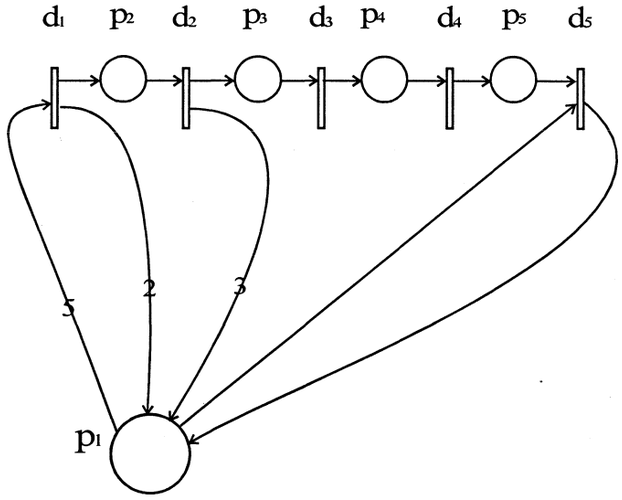


Рис.3.7. К определению 10

Определение 11.

Полным циклом называется часть максимального цикла, не содержащая мертвых переходов при произвольной начальной разметке ресурсной позиции, а также позиций, инцидентных только этим переходам.

Определение 12.

Полным классом циклов ресурсной позиции назовем множество, состоящее из полного цикла и всех циклов, являющихся его подмножествами.

Заметим, что если переход мертв, но инцидентен ресурсной позиции, это однозначно говорит об ошибке функционального описания ресурса. Нахождение меток в нересурсных позициях говорит об использовании некоторой части данного ресурса при обслуживании заявок на соответствующей стадии. Если сеть построена корректно, то общее число таких меток должно соответствовать числу обслуживаемых заявок с привлечением данного ресурса, и может измениться лишь при поступлении новой и по окончании обслуживания старой заявки. Такие события, в терминах рассматриваемого подмножества сетей, описываются срабатываниями начала и конца одного из полных циклов, — метки не могут возникать внутри цикла. Тогда можно сформулировать первое условие корректности функционального описания ресурса.

Теорема 3.

Общее число меток в нересурсных позициях равно разнице числа срабатываний начал и концов всех полных циклов.

Доказательство:

Исключим ресурсную позицию и все инцидентности к ней. Тогда, по построению, множество всех оставшихся переходов и позиций, есть подмножество автоматной подсети алгоритма, причем оно включает терминальные переходы. По свойству автоматной СП такое подмножество консервативно. Это означает, что сумма меток в этих позициях может измениться только при срабатывании терминальных переходов. Нетрудно видеть, что последние являются началами и концами полных циклов. Так как множество оставшихся элементов — ординарная СП, то срабатывание любого начала добавляет, а срабатывание любого конца извлекает только 1 метку из нее. При нулевой начальной разметке нересурсных позиций это эквивалентно содержанию теоремы. □

Полный класс циклов можно рассматривать как один цикл, в котором захваченная часть ресурса изменяется с каждой инцидентностью ресурсной позиции. Поскольку временные задержки исключены, то нулевая разметка всех позиций, исключая ресурсную, соответствует равенству текущей разметки ресурсной позиции, ее начальной раз-

метке. Семантически — это возвращение ресурса при отсутствии обслуживаемых заявок. Отсюда второе условие правильности назначения кратности инциденций.

Теорема 4.

Пусть живая СП: N = {P, D, F, H, Mo} является полным классом циклов:

P = (pi, p_2,..., pn), Pi —ресурсная; D = (d_b d_2,..., d_n); M_o = (m_0, 0,..., 0),

где п — длина полного цикла (натуральное).

Тогда для того, чтобы разметка Mi = (mi, 0,..., 0), mi #mo, не была достижима из Mo, достаточно, чтобы выполнялось:

$$YF(_{P\{};d_i) = \underset{z=1}{Y_l^{H}(di;i\)'} \qquad I=ï \qquad n \qquad n$$

то есть равенство суммы кратностей прямых и обратных дуг в пределах полного цикла.

Доказательство:

По определению 11 все элементы полного класса входят в полный цикл. Так как полный цикл — по определениям 6 и 11 частный случай простого цикла, то он не может дважды включать один и тот же элемент. Тогда он и не может иметь разветвлений в позициях (конфликтов), что соответствует повторному включению такой позиции в цепочку цикла. При Mo готово к срабатыванию только начало цикла. При Mi число срабатываний начала и конца равны (по теореме 3). При отсутствии конфликтных позиций (по условию теоремы), это означает последовательное срабатывание всех переходов, поскольку сеть жива. Так как все инцидентности нересурсных позиций единичны (автоматная подсеть алгоритма), число срабатываний всех переходов будет одинаковым.

Число изъятых из pi меток: $$\underset{z=l}{\overset{n}{}} \Pi\ = z \bullet {}^{\wedge}F(p\ ;dj),$$

число добавленных в pi меток: $$\underset{Z=1}{\overset{n}{}} = z \qquad ; p\ )\ '$$

где z — число срабатываний одного перехода.

С учетом условия теоремы П1= П 2 , ДЛЯ любого Z. Но тогда, если Mi^Mo , ТО И Щ* П 2 , что невозможно, следовательно Mi не достижима из Mo, что и требовалось доказать. □

Поскольку путь любой метки во фрагменте 3-го типа — это полный цикл, содержание теоремы 4 является одним из необходимым условий корректности установленных

ресурсных отношений. Однако оно может выполняться и для некорректной разметки, если установлен такой порядок захвата и возвращения ресурса, что возвращение ресурса следует раньше, чем захват, как например на рис.3.8.

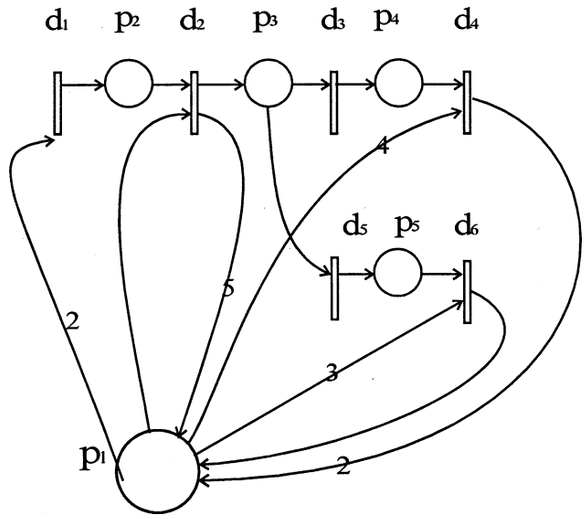


Рис.3.8. Пример некорректного назначения кратности инцидентных дуг

Кратности инцидентностей ресурсной позиции удовлетворяют условию теоремы 4, однако количество ресурса, захваченного процессом при срабатывании dz соответствует трем меткам, а возвращенного — пяти. Таким образом, число меток в ресурсной позиции может превысить начальную разметку, что, очевидно не верно. Отсюда второе необходимое условие корректности кратностей инциденций: разметка ресурсной позиции в произвольном состоянии сети M(pi) не больше начальной Mo(pi): Mo(pi) > M(pi).

Рассмотрим дискретную числовую ***функцию разности кратностей*** инциденций ресурсной позиции pi на множестве номеров переходов полного цикла:

$$K = \wedge N) = Y,m_{Pl},d_i) - md_{i,Pl})). \qquad (3.7)$$

где K — функция разности кратностей,

N — номер перехода в некотором полном цикле.

Теорема 5. Если в корректной сети (3-го типа) функция разности инциденций любого полного класса циклов неотрицательна, то для любого состояния сети текущая разметка ресурсной позиции не превышает начальную:

$$(K = 4/(N)>O) => (Mo(pi)>M(pi)), \qquad (3.8)$$

Доказательство: В корректной сети отсутствие меток в нересурсных позициях говорит о неизменности текущей разметки M(pi) по сравнению с Mo(pi) (теорема 3). По-

явление каждой новой метки в N-й (N^0) позиции полного цикла говорит о том, что произошло последовательное срабатывание первых его N переходов. При этом разметка ресурсной позиции M(pi) изменилась до M'(pi) на величину AMN:

$$AM_N = X(F(p_l,d_i)\text{-}^{M}(^{d}{}_i,p_l)) = \sum_{Z=1}^{K}(N)\ ' \qquad (19)$$

причем: $M'(p]) - M(pi)\text{-}K(N).$

Вообще в нересурсных позициях может быть по mN меток (ши = 0,1,2,...), которые изменили начальную разметку ресурсной позиции с Mo(pi) до M(pi) на величину:

$$AM=\hat{}m_N K(N)\ ' \qquad \sum_{W=1}^{N,,} \qquad (^{3Л0})$$

где N_n — общее число непустых нересурсных позиций в сети.

По условию теоремы $K(N) > 0$, следовательно для любого N *и* mN

$$(K(N) > 0) \text{ А } (AM > 0) \Rightarrow ((\text{Mo(Pl)} > \text{M (p0)},$$

что и требовалось доказать. □

Таким образом положительность функции разности кратностей может служить одним из критериев корректности сети.

Рассмотрим соотношение величин начальной разметки и кратностей инциденций. Если величина начальной разметки недостаточна для срабатывания всех переходов некоторого полного цикла, то в нем содержатся мертвые переходы, и сеть некорректна. Функционально это соответствует недостаточности ресурса для обслуживания единственной заявки по некоторой последовательности стадий обслуживания. Сформулируем еще одно необходимое условие корректности задания ресурсных отношений.

Теорема 6.

Функция разности кратностей инциденций любого живого полного цикла не превышает разности начальной разметки ресурсной позиции и кратности одноименной обратной инциденции:

$$d_N: K(N) < (M_0(pi)\text{-}H(d_{№,P1})), \qquad (3.11)$$

где pi-ресурсная позиция,

TV-номер перехода в полном цикле.

Доказательство:

Пусть для некоторого перехода dy выполняется противоположное условие:

$$K(N) > (\mathrm{Mo}(pi) - \mathrm{H}(dy, pi)). \quad (3.12)$$

Рассмотрим случай, когда сеть имеет состояние, соответствующее единственной ненулевой позиции среди нересурсных, и эта позиция прямо инцидентна переходу dy.

По определению функции разности кратностей инциденций имеем: для любого предшествующего перехода :

$$K(N-1) = Y(F(P\hat{}di) \sim H(d_{i>Px})) > \quad (3.13)$$

$$Z = 1$$

тогда для перехода dy :

$$N$$

$$Z=1$$

$$= \overset{AM}{\underset{Z=1}{X}} (F(P1.\ di) - H(d_t, {}_{P1})) + F(P!, d_N) - H(d_{N,}\ p,)\blacksquare \quad (3.14)$$

Подставим (3.13) в (3.14):

$$K(N) = K(N-1) + \mathrm{F}(p_b\ dy) - \mathrm{H}(dy, pi).$$

С учетом предположения (3.12) получаем неравенство:

$$\hat{}-/> + \mathrm{F}(pi,dy)-\mathrm{H}(dy,pi) > \mathrm{M}_0(pi)-\mathrm{H}(d_{\#},pi). \quad (3.15)$$

Так как метка в нересурсных позициях единична, в работе только один полный цикл,— используем рассуждения из доказательства теоремы 5 подставив (3.9) в (3.15), и решим неравенство относительно F(pi, dy):

$$\mathrm{F}(p_{1?}\ dy) > (\mathrm{Mo}(pi) - \mathrm{A\ My./}), \quad (3.16)$$

где A My./ — уменьшение разметки pi при срабатывании всех переходов полного цикла до TV-го, не включая dy.

Правая часть неравенства (3.16) представляет собой разметку ресурсной позиции Pi для рассматриваемого состояния сети. Тогда в силу общих правил функционирования СП неравенство (3.16) — условие мертвости перехода dy:

$$\mathrm{F}(pi, dy) > \mathrm{Mi}(pi). \quad (3.17)$$

Если же состояние сети отличается от рассматриваемого, то в силу теоремы 5 и

соотношения (3.9) ее разметка:

$$M_2(pi) < Mi(pi),$$

и условие (3.17) остается верным. Но по условию теоремы любой переход жив, следовательно верно противоположное (3.12) условие теоремы (3.11). □

Еще одним условием корректности фрагмента типа 3 является отсутствие взаимных блокировок между параллельными процессами, что в терминах СП определяется как невозможность попадания в тупиковые состояния. В условиях случайного характера срабатываний переходов возможна ситуация, когда при начальной разметке Мо, удовлетворяющему условию (3.11) теоремы 6, возникнет ситуация когда Мо уменьшится на столько, что ни в одном из полных циклов не сможет сработать ни один переход (рис.3.9а). Во фрагменте на рис.3.9б такая ситуация не возможна. В п. 3.1.3 описан достаточно общий критерий определения таких состояний на основе решения матричных уравнений состояния СП. Однако можно использовать и специфические особенности рассматриваемого класса сетевых моделей.

Теорема 7.

Если функция разности кратностей K(N) неотрицательна и возрастает только после нулевых значений: $(K(N_t) > K(Ni\ -1))$ A(/L(N;) = 0), то в живой СП тупики отсутствуют при любой допустимой начальной разметке Мо .

Доказательство:

Если бы это было не так, то существовала бы такая достижимая разметка из Мо, при которой два или более переходов .. .ф,.. .ф,... не могут сработать, причем:

$$\begin{aligned} &M(P_0) < F(P_0,di)\text{-}, \\ &M(P_0) < F(P_0,dj). \end{aligned} \qquad (3.18)$$

Но по условию теоремы, — возрастание $K(Nf)$ возможно при $K(N\backslash — 1) =0$, что соответствует возвращению всех меток в P_o. Тогда при выполнении (3.18) СП и без того мертва, что противоречит условию теоремы. Следовательно неверно (3.18), что и требовалось доказать.

В данном пункте сформулирован ряд необходимых условий корректности сетевого описания УК, основанных на использовании топологических ограничений рассматриваемого класса сетей, точнее его фрагментов. Формулировка достаточных условий корректности затруднена, если вообще возможна, поскольку в каждом конкретном слу-

чае сетевая модель строится на основе большого многообразия алгоритмов работы, моделей внешнего окружения, структур УУ УК, операционных систем, интерфейсов и т.д. Поэтому отыскание общего достаточного условия правильности вряд ли возможно. Однако использование предложенных необходимых условий позволит исключить значительное количество ошибок при отладке моделей.

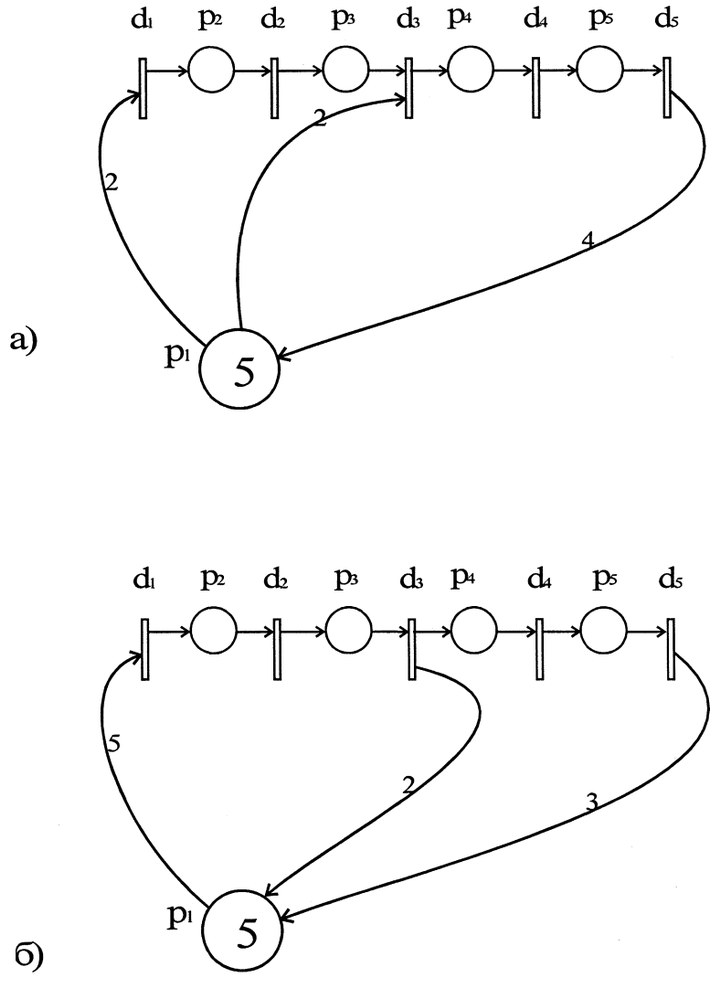


Рис.3.9. К пояснению взаимных блокировок между параллельными процессами

Примем без строгого доказательства утверждение о том, что дополнение СП, удовлетворяющей топологическим ограничениям фрагмента 2, ресурсными позициями так, что выполняется условие теорем 3-е-7, не нарушает свойств живости, терминальности, детерминированности и ограниченности при допустимой начальной разметке их, на основе следующих соображений:

1. Живость не может нарушаться добавлением некоторой ресурсной позиции, так как каждая из них входит только в живые, нетупиковые циклы.

2. Терминальность не нарушается, так как прекращение поступления меток соответствует прекращению срабатываний начал полных циклов. Поскольку от каждой метки, попавшей в нересурсную позицию сработает каждый его переход, по теореме 3 восстановится начальная разметка каждого полного цикла. Так как выполнение любого действия в УК связано с захватом ресурса, то, за исключением терминальных позиций, вся СП покрывается полными циклами, и восстановит свою начальную разметку.
3. Детерминированность не может нарушаться, так как появление дополнительных инцидентностей у конфликтных переходов от ресурсных позиций и установка приоритетов разрешает любой их конфликт.
4. Ограниченность не нарушится, так как всякая новая обратная инцидентность к ресурсной позиции входит в некоторый ограниченный полный класс циклов.

Подытожим действия по анализу сетей на основе топологических ограничений:

1) декомпозиция-выделение фрагментов трех типов по указанным выше правилам;
2) проверка фрагментов первого типа на принадлежность к классу автоматных СП;
3) проверка совокупностей внутренних путей подсети ВО во фрагментах типа 2 на принадлежность к автоматным СП;
4) проверка фрагментов типа 2 на принадлежность к простым чистым СП и по критерию (теорема 2) на корректность ;
5) проверка фрагментов третьего типа на корректность (теоремы 3 4- 7).

3.1.3. Метод уравнений состояния сети.

Метод рассмотрен, например, в [66]. Основная идея метода — описание различных состояний СП и динамики их смены рекуррентными уравнениями, связывающими векторы исходной и результирующей разметок с матрицами инцидентности и управляющим вектором, отражающим правило возбуждения переходов. Для обычной СП такое уравнение имеет вид:

$$Mk=M_k.i + A*U_k, \qquad (3.19)$$

где к — номер такта работы СП,

М к — разметка СП после к тактов работы (Мо — начальная разметка),

А* — транспонированная матрица добавления меток в позиции :

$$A = V\text{-}O^*, \tag{3.20}$$

где V — матрица обратных инциденций,

O* — транспонированная матрица прямых инциденций,

U к -— управляющий вектор, компоненты которого равны 1, если условия возбуждения данного перехода выполняются в k-м такте работы СП:

$$Vi:(m(p>) > F(pi,dj)=>(uj=l). \tag{3.21}$$

Рекурсивные уравнения позволяют описывать разомкнутые СП с внешним источником меток. В этом случае уравнение (3.19) дополняется вектором поступающих меток Wк :

$$Мк = M_к\text{-}1 + A*U_k + Wk . \tag{3.22}$$

Рекурсивные уравнения, описывающие функционирование СП позволяют использовать методы линейной алгебры для оценки основных алгоритмических свойств СП. Например, можно вычислить p-инвариант X сети, как решение матричного уравнения

$$AX = 0 . \tag{3.23}$$

Из уравнений состояния СП можно получить соотношение

$$X*M = X*M_0, \tag{3.24}$$

где M — любая достижимая разметка.

Отсюда вытекает смысл компонент вектора p-инварианта — это “весовые” коэффициенты меток в позициях, характеризующие их распределение в сети. В [66] доказано, что если все компоненты p-инварианта положительны, то СП ограничена. Такой p-инвариант называется p-цепью, а СП — инвариантной. Кроме того можно определить t-инвариант Y как решение матричного уравнения:

$$A*Y = 0. \tag{3.25}$$

Для t-инварианта верно соотношение :

$$Мо=Мо+A*Y, \tag{3.26}$$

из которого видно ,что последний характеризует свойство сети неизбежно возвращаться в исходное состояние Мо, то есть при Y = 0 СП устойчива. Аналогично p-цепи опреде-

ляется t-цепь. Ее наличие является достаточным условием живости СП при любой ненулевой начальной разметке.

Одним из результатов рассмотрения уравнений состояния СП является возможность достижимости из начальной — разметки М, как решения системы матричных уравнений:

$$BM = BM_6', \qquad (3.27)$$

$$[OM=OE-E,$$

где О* — транспонированная матрица обратных инцидентностей,

Е — единичный вектор,

В — матрица инвариантов, получающаяся объединением векторов-строк фундаментальной системы решений уравнений (3.23).

Решениями системы (3.27) являются тупиковые разметки. Методы решения систем уравнений, подобных (3.23), (3.25), (3.27) изложены, например в [4].

Исследование рассматриваемых в работе СП методом уравнений состояния возможно при допущениях, исключающих специальную логику ее работы (п.3.1.1), и к рассмотрению принимались бы СП, функционирующие по правилам обычных. В соответствии с соображениями, изложенными в п.3.1.1, метод целесообразно использовать с целью анализа на ограниченность — выражение (3.23) и на наличие тупиковых разметок — выражение (3.27). Преимуществами метода являются универсальность, то есть применимость к обычным СП без топологических ограничений, а так же удобная, компактная математическая, формулировка, облегчающая практическую реализацию.

Однако возможности данного метода ограничиваются тем, что в нем сформулированы лишь достаточные условия, например, не для всякой ограниченной СП существует полная р-цепь и т. д. Кроме того не каждая из проблем, на целесообразности решения которых мы остановились в п.3.1.1, разрешимы на основе данного метода. Таким образом, остановимся на использовании метода, как проверочного на наличие свойства ограниченности и тупиковых разметок.

3.1.4 Метод построения дерева достижимости.

Наиболее общим подходом к анализу СП на различные свойства является рассмотрение множества ее достижимых разметок (маркирований) от начальной R(N,Mo),н соответствующего ему множества последовательностей срабатывающих переходов или свободного языка СП L(N). Любая достижимая из Mo разметка соответствует некоторому состоянию СП и моделируемого ею объекта, а соответствующая ей последовательность срабатывающих переходов (слово L(N)) — предыстории этого состояния. Множества R(N,Mo) и L(N) полностью характеризует поведение системы в терминах СП, их анализ позволяет получить разрешимость практически всех основных алгоритмических проблем и оценить ряд количественных показателей функционирования СП.

Обычно множества R(N,Mo) и L(N) представляет в виде направленного графа, вершина которого соответствует одной разметке, а наборы дуг — последовательностям срабатывающих переходов, в результате которых из одной разметки получается другая. Такой граф получил следующие названия, которые будем считать эквивалентными:
- граф разметок;
- граф достижимых разметок;
- дерево достижимых разметок;
- дерево достижимости.

В общем случае граф достижимых разметок может быть бесконечным, его вид зависит от принятой концепции работы СП. Для обычных СП в настоящее время принята асинхронная модель с произвольной кратностью дуг, в которой возбужденные переходы могут срабатывать независимо и параллельно [34].

Исследование графа достижимых разметок не всегда удобно, так как при наличии неограниченных позиций такой граф бесконечен, что не позволяет получить гарантированного ответа на вопрос о наличии того или иного свойства. В [34] предложено использовать для таких целей граф несколько иного типа, в котором конечными вершинами является такие разметки M, что выполняется одно из условий:

M =M, или

M>M,

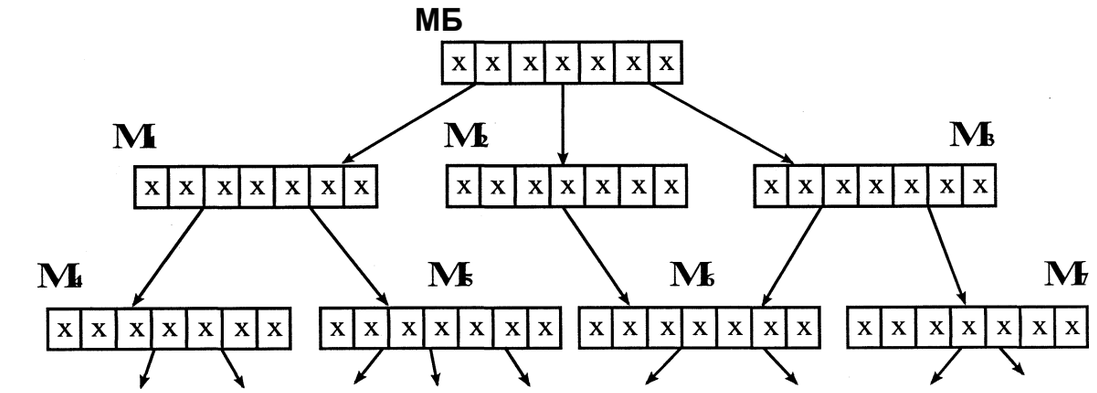
где M — одна из вершин, уже полученных ранее.

Такой граф получил название полного покрывающего дерева сети. Для любой СП он конечен [34], его анализ дает возможность разрешения основных алгоритмических проблем.

Как было указано выше, опыт использования СП для функционального описания различных дискретных динамических систем показал высокую трудоемкость исследования их методом построения и анализа графа достижимых разметок или полного покрывающего дерева сети. Даже для СП с числом вершин в несколько сотен, объем информации занимаемой таким графом, совместно с необходимой служебной информацией, а также требуемое быстродействие может оказаться не под силу массовым моделям современных персональных компьютеров. Поэтому область применения такого метода ограничивается анализом СП сравнительно небольшой размерности (например СП частных алгоритмов) и умозрительными (мысленными) экспериментами для доказательства тех или иных утверждений.

Предлагаемое расширение СП, как было указано выше, имеет значительные отклонения от концепции обычных СП. Возникает вопрос о возможности построения покрывающего дерева для данного класса СП, и о способе построения, если оно возможно. В п.3.1.1 рассматривалось понятие разметки временной СП, в которой меткам приписывалось пассивное состояние в позиции, под разметкой понималось общее число меток в ней. Такое представление удобно для количественной характеризации СП, но оно не отражает в полной мере состояния временной СП, как это имеет место в обычных СП, так как не ясно сколько, каких меток, на какое время задержаны. В этом смысле более удачным и естественным представляется концепция мгновенного поглощения при срабатывании и “нахождения” задержанных меток в переходах. При этом понятие разметки позиции расширяется, и включает не только те метки, которые выданы уже входными переходами в позицию, а и те, которые неминуемо будут выданы ими через некоторое время. Таким образом вектор разметки СП трансформируется в двухмерную матрицу, одно измерение которой, как и ранее — номер позиции — “пространственная” координата. Другое измерение — число тактов работы СП до выдачи меток в позицию — “временная” координата. В клетках матрицы — суммарное число меток, выдаваемых через 0,1,2,...,п тактов работы СП в сети переходами в позицию. Размерность матрицы — n x m, где п — число позиций СП, m — максимальная из задержек по всем перехо-

дам СП. Пример, поясняющий трансформацию представлен на рис.3.10 а и б). Имеется ввиду, что вертикальное измерение матриц — время в тактах временной базы сети.



а)

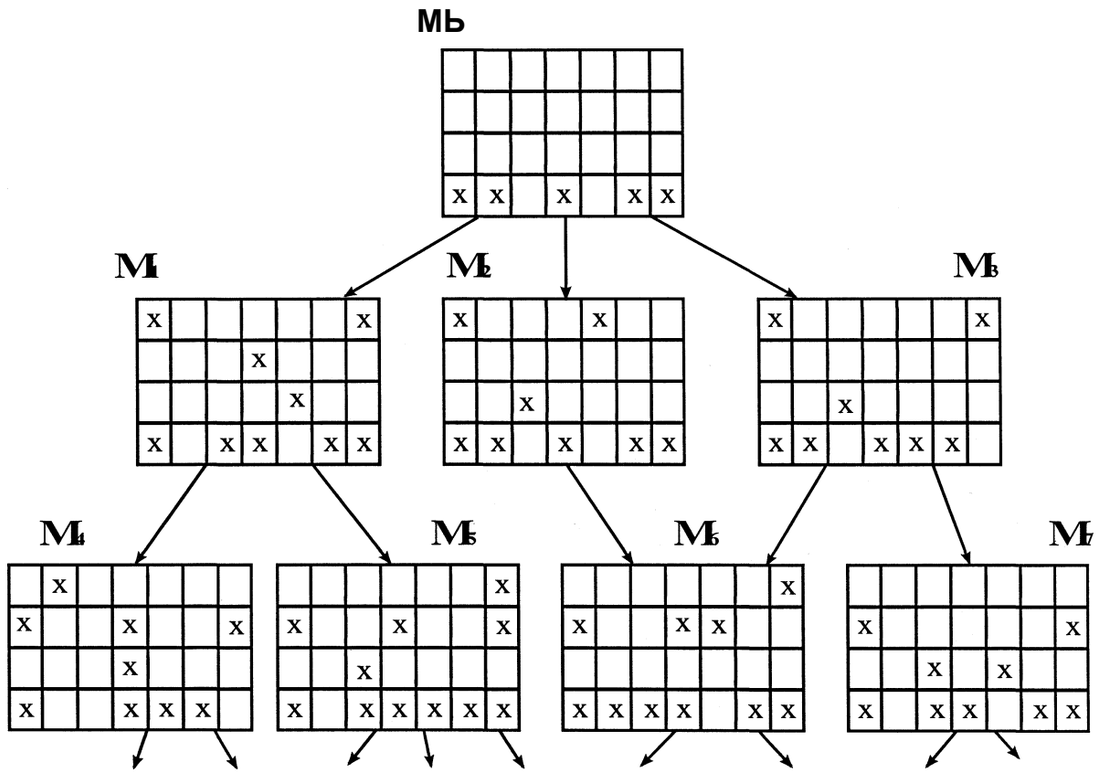


Рис.3.10. Пример структуры покрывающих деревьев:

а) для обычной СП, б) для временной СП

Соотношения между вершинами аналогичны соотношению вершин дерева обычной СП. Две разметки равны, если равны все элементы матриц. Одна разметка больше другой, если каждый элемент одной не меньше, и по крайней мере 1 — строго больше соответствующего элемента в другой матрице. Таким образом, для расширенного понятия разметки временной СП сохраняются все математические отношения, действительные для разметки обычной СП. Такой вывод можно объяснить по-другому, используя понятие взаимного преобразования обычной СП и синхронной временной путем замены временных переходов последовательностью обычных переходов и позиций (рис.2.16). Такая трактовка разметки СП дает возможность выделить эквивалентные состояния временной СП — равенство разметок, и такие, которые свидетельствуют о неограниченности СП — превышение разметки вершины-потомка над разметкой вершины-родителя. Однако конечность такого дерева не удается доказать, что не дает гарантий на получение ответов на вопросы об ограниченности, потенциальной живости и достижимости некоторой разметки для временной СП путем его построения.

Наличие приоритетов, как легко видеть, может лишь уменьшить число возбуждаемых переходов в каждом состоянии СП, и, как и управляющие функции ЖГСМ, не оказывает влияния на форму представления состояния СП.

Таким образом, можно сделать вывод о возможности использования метода для рассматриваемого класса СП при их относительно небольшой размерности или простой топологии.

3.2. Методы количественного анализа.

Априорные количественные характеристики функционирования некоторого оригинала в принципе можно получить на основе моделей двух основных типов:

1. Математических выражений, связывающих в систему уравнений различные характеристики объекта, потока заявок, алгоритмов работы и т. п.
2. Имитационных моделей, основанных на воспроизведении в том или ином виде функционирования оригинала в заданных условиях.

Модели первого типа обычно требуют значительно больших упрощающих допущений в том смысле, что получение сколь-нибудь доступных для анализа выражений возможно лишь для достаточно простых моделей, при учете относительно небольшого числа воздействующих факторов. При этом достоверность полученных результатов для сложных систем обычно невелика. Часто решение математического выражения в явном виде получить очень сложно или вообще невозможно. В таких случаях прибегают к численным методам нахождения приближенных решений.

Применительно к СП, некоторое распространение получил метод разложения СП на более простые подсети, позволяющий получить аналитические зависимости (рассмотрен в п.3.2.2).

Модели второго типа обычно реализуются как программы ЭВМ, в которых некоторым переменным сопоставлены характеристики объекта, потока заявок и т. п., а моделирующий алгоритм отражает алгоритм функционирования объекта. Возможности данных моделей по отношению к сложным оригиналам более широки. Имитационные модели реализуемы при значительно меньших упрощающих предложениях и дают вполне приемлемые по точности результаты. Преимущества имитационных моделей описаны в [25,67]. К недостаткам таких моделей можно отнести довольно большие затраты на написание программ, зависимость точности результата от времени моделирования, трудности применения классической теории статистики к обработке результатов из-за невозможности проведения многократных независимых испытаний (например моделирование 10^6 раз ЧНН работы УК). Тем не менее такие модели являются основным средством априорного исследования сложных систем.

3.2.1. Имитационное моделирование.

В общем случае, имитационная модель может быть реализована как детерминированная (например метод “Динамо”), так и статистическая (метод “Монте-Карло”) [62,92]. Предлагаемая методика сетевого описания УК рассматривает сетевую модель, как совмещение стохастической (ВО) и детерминированной части (алгоритм, ресурсы). Такой подход определяет имитационную модель, как статистическую, в которой необходимо моделировать случайные величины к функции ГСМ (ГСЧ) и ЖГСЧ (ЖГСМ). Общий принцип моделирования можно представить в виде схемы на рис.3.11.

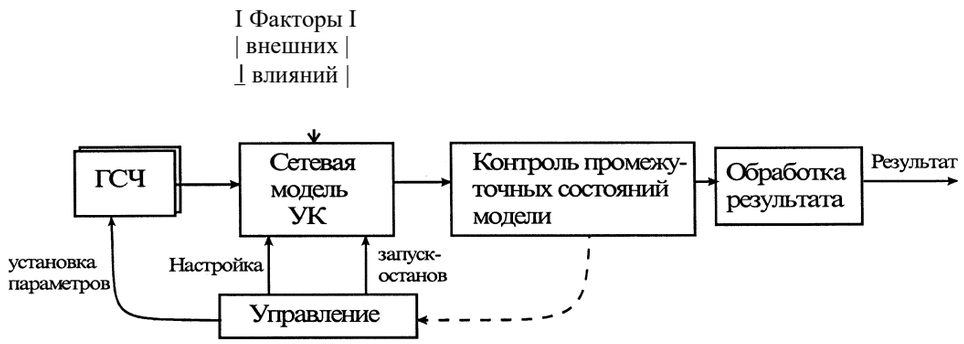


Рис.3.11. Схема имитационного моделирования УК

СП естественным образом задает формат исходных данных и структуру моделирующих программ в удобной, наглядной форме. Топология СП, математически задаваемая матрицами прямых и обратных инцидентностей, легко отображается двухмерными массивами (структурированными списками). Разметка, так же легко отображается переменной-вектором (простым списком). Проверку условий возбуждения переходов и их работу можно реализовать векторными операциями. Для генерации случайных чисел можно использовать встроенные функции ГСЧ с равномерным законом распределения, имеющиеся во многих языках программирования. Требуемый закон получается применением к такому числовому ряду математических операций. При отсутствии встроенного ГСЧ или ее низком качестве, можно использовать подпрограммы [29]. Временные задержки переходов можно реализовать присвоением переходам отдельных переменных векторов (списков), отображающих логику работы, представленную на рис.2.16, например сдвигом вправо содержимого подпозиций. Для тех же целей можно использовать двоичное преставление числа и булеву операцию сдвига вправо. Приоритетность СП

легко реализовать установлением обработки переходов не в порядке номеров, а по специальному списку номеров, составленному с учетом приоритетов переходов.

Как видно, организация функционирования предлагаемых СП в имитационной модели на основе языков программирования высокого уровня не вызывает особых сложностей. При использовании специализированных языков, (GPSS, Симула и др.), ориентированных на моделирование, отдельные элементы СП могут быть представлены объектами данных языков. Вопросы их использования рассмотрены в [62,68].

Модель УК должна обеспечить возможность изменения структурных и функциональных параметров, а так же получение некоторого результата, обеспечивающего возможность оценки по критерию. Для указанных принципов построения модели это означает, что перенастройка структурных параметров сводится к изменению матриц инцидентностей, а функциональных—к преобразованию векторов разметки, временных задержек и приоритетов, параметров законов распределения ГСМ и ЖГСМ. При вынесении этих данных в отдельные файлы настройки, возможно построение универсальной моделирующей программы для рассматриваемых сетевых моделей.

Получение результатов моделирования сводится прежде всего к анализу разметки. В данной сетевой модели разметка является функцией модельного времени. Представляет интерес состояние разметки, как по окончании процесса моделирования, так промежуточные состояния — в процессе работы модели. Аппарат СП допускает реализацию различных вычисляющих, логических функций и запоминания (п.4.2.2). Следовательно, имеется возможность оценки не только результирующего состояния модели по сложившейся разметке, но и подсчета числа интересующих ситуаций путем введения в модель вспомогательных, вычисляющих элементов СП. Более подробно данный вопрос рассмотрен в п.4.2.2.

Метод имитационного моделирования позволяет расширить рамки исследований путем введения различных функций влияния (помехи, случайные ошибки), рассмотрение нестационарных режимов работы (изменение параметров ГСЧ в процессе функционирования) и т.д. Экспериментальные исследования показали, что метод вполне приемлем.

Таким образом можно сделать вывод о принятии метода имитационного моделирования для количественного анализа рассматриваемых сетевых моделей как основного средства исследования.

3.2.2. Метод разложения на инвариантные и состоятельные подсети

Метод описан, например в [62], и предполагает количественный анализ временных СП при следующих ограничениях:

1. Сеть должна относиться к подклассу ординарных, чистых СП, то есть :

> = PxZ>->{0, 1};

<Я = £>xP->{0, 1};
\fdj є D: (F(dj) n H(dj) = 0}.

2. Временные задержки присвоены позициям, постоянны и не зависят от времени.
3. Сеть имеет циклическую структуру — каждый элемент входит в некоторый цикл.

Разметка, в терминах данного метода, называется переменной заряда сети :

где $q_i(t)$ — разметка i-й позиции в момент времени t,

n — число позиций в сети.

В методе рассматривается, так называемый, вектор тока сети :

$$h$$

где v(t) — вектор срабатываний переходов, компоненты которого ipk(t) равны числу срабатываний k-го перехода к моменту времени t,

ik — средняя частота срабатываний k-го перехода за интервал времени At = t - to:

m — число переходов в сети.

Задержки в позициях задаются матрицей задержек :

O 0 O
z_2 o - o
O 0 ••• Zn

Отношения инцидентности задаются матрицей инциденций C:

$$C = V - O^*,$$

где V — матрица обратных инциденций,

O* — транспонированная матрица прямых инциденций.

Основные соотношения, используемые для получения количественных характеристик действительны для установившегося режима функционирования сети, при котором токи постоянны, а суммарный заряд ограничен:

$$CI = 0 , (I > 0); \quad (3.28)$$

$$\backslash j_s Q(t_0) > J_s Zn \backslash , \quad (3.29)$$

где $IT\ I = G'$ — множество векторов J_s таких, что t — инвариант сети J_o может быть представлен как линейная комбинация J_s с неотрицательными коэффициентами:

$$j'c = 0; j\backslash = У\ h\ j\backslash , (hi > 0).$$

5

Такое множество G' получило название генератора сети. Для нахождения решения уравнений (3.28) и (3.29) предложено разбиение сети на элементарные состоятельные и элементарные инвариантные подсети, покрывающие анализируемую СП. Элементы их определяются генератором. В качестве состоятельных удобно выбирать синхронизационные (маркированные) графы, а в качестве инвариантных — автоматные СП. Такой выбор обусловлен простотой нахождения вектора Jo, например, при указанных ограничениях, в случае автоматных инвариантных подсетей решением системы:

$$ГС = 0$$

является вектор Jo = (1, ... , 1).

Каждая выделенная подсеть замкнута и циклична, что определяется условием равенства нулю суммы векторов — строк матрицы инциденций С, соответствующих позициям, вошедшим в подсеть. Число подсетей должно быть минимально. Разбиение на инвариантные подсети определяет генератор G'. Тогда решение уравнения (3.29) при известных задержках даст требования к начальной разметке, под которой обычно подразумевают ресурсы, или позволит определить максимальные токи, под которыми обычно понимают интенсивности заявок и т.д., в зависимости от постановки задачи.

Аналитические зависимости, связывающие интенсивности потоков заявок, ресурсы, конфигурацию обслуживающего устройства и алгоритмы его работы, — безусловно наиболее желаемый результат сетевого моделирования. Возможность получения таких зависимостей — главное достоинство рассматриваемого метода. Однако применение его для исследования сетевых моделей УК затруднено по следующим причинам.

1. Первое ограничение, предлагаемое данным методом, не может быть выполнено, так как ординарность СП вносит неадекватность в описание многих ресурсов, так как невозможно отразить их частичный захват. Требование чистоты СП может быть реализовано детализацией частных алгоритмов моделируемых переходами. Это, в свою очередь, настолько усложнит сетевую модель, что теряется всякая наглядность модели, многократно возрастает вероятность ошибок, математические операции, описанные выше, становятся трудновыполнимыми, если их вообще возможно реализовать.

2. Второе ограничение требует присвоения временных задержек позициям, что противоречит принятой концепции построения модели, вкладывает совершенно иное семантическое содержание в элементы СП. Формальная взаимозамена позиций и переходов [59] невозможна, так как сеть теряет живость, и начальную разметку.

3. Третье ограничение, как было указано в разделе 2, также не выполняется.

4. В подклассах обычных СП невозможно моделировать случайный поток заявок с заданными параметрами и случайную временную задержку, что значительно снижает достоверность результатов расчета.

5. Реализация метода для сложных СП большой размерности, особенно на этапах выделения минимального множества покрывающих инвариантных (состоятельных) подсетей и решения уравнения (3.29) чрезвычайно трудоемка, требует перебора и анализа большого числа вариантов.

Как видно из методики построения сетевых моделей УК, отображение алгоритма, подробность которого задается SDL — диаграммой, описание ВО и ресурсов, предполагает сеть достаточно большой размерности (с числом позиций или переходов от нескольких десятков до нескольких сотен). В силу приведенных рассуждений можно сделать вывод, что применение метода к общей сетевой модели невозможно. Однако он может использоваться для анализа некоторых частных алгоритмов, с целью получения некоторых исходных данных общей сетевой модели.

ВЫВОДЫ

В соответствии с поставленной в работе задачей, в разделе 3 проведен поиск методов алгоритмического и количественного анализа сетевых моделей, методика построения которых разработана в разделе 2. По сложившемуся в теории СП подходу исследование алгоритмических свойств проведено в три этапа:

1) выделение и формальное определение тех свойств сети, которые имеет смысл анализировать,
2) решение вопроса о принципиальной возможности распознавания выделенных свойств, для данной сети или класса сетей,
3) нахождение оптимального способа распознавания тех свойств сети, для которых это возможно.

Для определения состава анализируемых свойств дана смысловая интерпретация основных алгоритмических проблем СП в соответствии с содержательным смыслом сетевых моделей УК. Определено, что свойства *ограниченности, непротиворечивости (детерминированности), живости (потенциальной живости), терминальности* являются необходимыми условиями правильности построения сетевой модели и выбора ее динамических параметров, признаком конструктивности реализуемых УК алгоритмов. Сети, обладающие такими свойствами, названы *корректными.* В соответствии с принятыми расширениями сетевой модели свойство терминальности переопределено так, что множество переходов, порядок срабатывания которых определен функцией ЖГСМ, считается детерминированным.

Исследование разрешимости алгоритмических проблем, соответствующих данным свойствам показало, что проблема терминальности строго разрешима, а остальные разрешимы лишь частично, на основе рассмотрения разрешимости более широкой одноименной проблемы для обычной СП совпадающей топологии. Тем не менее, данный результат позволяет оценить свойства предложенного расширения и дает возможность использования результатов классической теории СП.

Для нахождения критериев верификации сетевых моделей использован тот факт, что методика их построения налагает существенные ограничения на топологию. Например, подсеть алгоритма является автоматным графом, что является прямым следствием автоматной природы языка SDL который транслируется в СП. Отсюда вытекает и

предложенный в работе прием дальнейшего разделения сетевой модели на фрагменты по функциональному признаку — функциональной декомпозиции. Фрагменты выбираются так, чтобы:

- подчиняться одному из известных топологически ограниченных подклассов,
- достаточно полно была представлять включающую его подсеть,
- выделенные фрагменты покрывали всю сетевую модель.

Предложено рассматривать автоматную подсеть алгоритма; ординарную, простую и чистую подсеть ВО, строго цикличные фрагменты подсети ресурсов. Несложные преобразования позволяют получить и автоматную подсеть ВО.

Идея выбора критериев состоит в том, чтобы сохранялась возможность его автоматического вычисления и определяемое им условие было признаком корректности фрагмента. Это позволяет упростить задачу верификации. Известно, что в классе автоматных СП разрешимы все основные алгоритмические проблемы, Проверку живости ее предложено реализовать путем инерации и определению сильносвязности или по критерию свободных СП. Для оценки цикличных фрагментов нет известных строгих критериев корректности поэтому сформулированы и доказаны пять теорем, которые могут служить таковыми.

Наряду с известными из теории СП, описанные приемы и критерии могут служить основой автоматической проверки корректности предложенных сетевых моделей. Рассмотрены также известные методы анализа СП — уравнений состояния сети и построения дерева достижимости. Определено, что они имеют ряд недостатков, исключающих их применение для анализа сложных сетевых моделей, однако они могут использоваться для анализа относительно простых моделей частных алгоритмов на этапе вычисления временных задержек переходов алгоритма.

Рассмотрение известных методов количественного анализа СП позволило установить, что альтернативы имитационному моделированию нет. Все принятые расширения СП легко реализуются языками программирования, метод нагляден имеет удовлетворительное быстродействие. Известный метод разложения СП на инвариантные и состоятельные подсети требует почти таких же ограничений, как и СМО, применим только к относительно несложным временным СП с задержками в позициях, и может использоваться только для определения временных задержек частных алгоритмов.

РАЗДЕЛ 4

АНАЛИЗ СЕТЕВЫХ МОДЕЛЕЙ И ОБОБЩЕНИЕ МЕТОДА МОДЕЛИРОВАНИЯ

4.1. Предварительный анализ модели на основе качественных методов

Процесс построения общей сетевой модели реального УК имеет сложный, многоступенчатый характер, что обусловлено сложностью оригинала и многообразием его возможных реализаций. Полностью автоматизировать этот процесс по-видимому нельзя из-за трудностей компоновки элементов графа СП и задания исходных данных по построению подсети УУ УК (ресурсных отношений). Естественно, что в таких условиях весьма возможно возникновение ошибок описания, связанных с невнимательностью и просчетами экспериментатора. Некоторый опыт построения и испытания подобных моделей позволяет выделить следующие классы ошибок :

1 .Ошибки, связанные с нарушением общих правил топологического строения:

- ошибочные инцидентности переход-переход И ПОЗИЦИЯ-ПОЗИЦИЯ,
- объявления неконфликтных позиций как ЖГСЧ,
- дублирование номеров переходов и позиций и т.п.

2.Ошибки, возникающие при определении матриц инцидентности по графу СП.

3.Ошибки, связанные с некорректным назначением начальной разметки ресурсных позиций, нарушающие работу СП.

4.0шибки описания ресурсов, приводящие к возможным взаимным блокировкам параллельных процессов (тупикам).

5.Ошибки построения графов СП, приводящие к нарушению их основных алгоритмических свойств: потенциальной живости, ограниченности, недостижимости тупиков, терминальности и т. п.

б.Ошибки назначения функциональных параметров модели:

- приоритетов,
- временных задержек,
- законов распределения ЖГСЧ.

7.Ошибки несоответствия топологии СП оригиналу, не нарушающие формальных требований к алгоритмическим свойствам СП.

Ясно, что прежде чем приступить к испытаниям сетевой модели УК, необходимо провести ее предварительный анализ с целью выявления возможных ошибок и, хотя бы частичного, подтверждения ее корректности. Все сказанное можно отнести и к сетевым моделям частных алгоритмов (п.2.1.2.).

Ошибки первого и второго типов можно устранить автоматизацией анализа графа СП и программной генерацией матриц инцидентностей по графу. Возможности современных графических сред со встроенными языками программирования позволяют эффективно строить и анализировать изображения графов СП практически любой размерности. Возможная реализация такой программы представлена в приложении Б.

Часть ошибок третьего и четвертого типа можно избежать при проверке выполнения условия теорем 6 и 7 (п.3.1.2). То же можно сказать об ошибках типа 5 и теоремах 2, 4 и 5 (п.3.1.2). Условия, базирующиеся на использовании топологических ограничений, удобно применить в процессе получения исходных данных моделирования. Проверку свойств готовой модели удобно произвести вычислением инварианта СП и решением уравнений состояния сети, рассмотренных в п.3.1.3.

Ошибки типов 6 и 7 трудно поддаются выявлению в СП большой размерности, если они не приводят к очевидным последствиям, которые можно заметить при рассмотрении промежуточных и конечной разметок СП при моделировании. Для облегчения их нахождения необходимо стараться формировать граф СП в возможно более структурированной форме, чтобы максимально использовать свойство наглядности СП, например путем разнесения частей чертежа графа в отдельные слои. Таким образом наиболее трудно выявляемые ошибки можно рекомендовать обнаруживать оценкой ожидаемого результата при пробном моделировании и визуальным анализом графа СП.

Уменьшить вероятность ошибок при назначении временных задержек переходов можно на этапе анализа и моделирования частных алгоритмов. Для многих из них граф, моделирующей СП, оказывается сравнительно небольшим (по отношению к графу СП общей модели), что позволяет применить для анализа такой модели наиболее универсальный способ — построение покрывающего дерева сети (п.3.1.4). Возможный подход к использованию этого метода показан в приложении А, где имеется блок-схемы и листинги двух программ (с пояснениями), предназначенных для построения и исследования покрывающего дерева (ПД) временной приоритетной СП (Turbo Pascal [75]-Delphi).

4.2. Анализ сетевых моделей на основе количественных методов

4.2.1. Общая концепция модели. Исходные данные

В соответствии с оценкой применимости различных методов для количественного анализа предлагаемых СП, произведенной в разделе 3, принято решение об использовании в качестве основного средства получения прогнозируемых характеристик функционирования УК — метода имитационного моделирования. В данном пункте рассматриваются общие вопросы построения таких моделей.

Исследование моделей УК затрагивает очень широкий круг вопросов, связанных с их функционированием в реальных условиях. Данная работа не ставит целью полноту охвата всех таких вопросов, и предполагает рассмотрение лишь наиболее значимых для оценки структур УК особенностей их работы, которые непосредственно связаны с обслуживанием вызовов и могут быть описаны в терминах SDL. Поэтому в общей концепции не будем рассматривать влияние на обслуживание заявок таких факторов: аварийные ситуации; аппаратурный и программный контроль, тестирование, самотестирование; самообучение, автоматическая реконфигурация; ошибки ПО; случайные помехи (сбои); прочие факторы. При необходимости, арсенал средств описания УК, рассмотренный в разделе 2 все таки позволяет учесть большинство указанных факторов, и они могут включаться в модель, при соответствующих целях моделирования. Возможные подходы к данной проблеме рассмотрены в п. 2.4.

Предлагаемые сетевые модели УК являются достаточно сложными, даже в случае общего описания, и при использовании упрощающих предположений. Рассмотрение принципов построения таких сетей в разделе 2 и методов их анализа в разделе 3 определило их основные особенности, а именно:

- сеть должна быть *временной,* временные задержки присваиваются переходам;
- сеть должна быть *приоритетной* и/или *ингибиторной',*
- сеть должна иметь средства для моделирования случайной временной задержки и других случайных событий — предложено реализовать посредством *ГСМ, ЖГСМ',*
- сетевая модель может включать *управляющую функцию,* которая устанавливает специальные правила для некоторого подмножества переходов;

- сеть *разомкнута,* то есть содержит терминальные элементы: не менее одной входной позиции (ГСМ), и не менее одного выходного элемента (позиции или перехода);
- сеть *связна,* то есть из каждой позиции в каждую существует путь вдоль или против инцидентных дуг;
- фрагменты сети, выделяемые по особым правилам, подчинены некоторым *топологическим ограничениям',*
- общая сеть, описывающая УК, естественным образом представима в виде *композиции нескольких подсетей,* сопрягаемых по переходам подсети алгоритма;
- сеть включает четко разграниченные *детерминированную и стохастическую* части, как подсети.

Принятие положения о необходимости детерминированности процессов в подсетях, описывающих алгоритм и ресурсы УК налагает требование на *синхронность* [34] функционирования общей сети. Это означает, что работа сети разбивается на такты, внутри которых определяется множество взаимно не конфликтующих, возбужденных переходов. Каждый переход из этого множества обязательно срабатывает в данном такте. Если в результате срабатывания части переходов появляются условия для срабатывания новых переходов, таких, которые не были возбуждены к началу такта, то они не прибавляются к указанному множеству и в текущем такте не срабатывают. Заметим, что под *конфликтующими* понимается такое множество переходов, что срабатывание одного (одних) из которых может исключить возможность срабатывания другого (других) переходов из этого множества при текущей разметке сети в такте. В неингибиторной сети конфликтующими являются переходы, имеющие общую входную позицию, разметка которой меньше суммы кратностей ее прямых инцидентностей. Тактирование сети требует решения вопроса о характере поступления и перемещения в ней заявок. Наиболее очевидными являются три возможных альтернативы:

1. Заявки поступают "порциями" за такт работы сети. Величина "порций" случайна и подчинена некоторому закону распределения. Продвижение заявок из позиции в позицию в переходах так же "порционно" — переход может изымать, выдавать и задерживать любое число заявок (любую порцию).

2. За такт работы сети может поступить не более одной заявки. Число тактов между поступлением двух соседних заявок случайно и подчинено некоторому закону рас-

пределения. Перемещение заявок из позиции в позицию в переходах может быть "порционным".

3. За такт работы сети может поступить не более одной заявки. Число тактов между поступлением двух соседних заявок случайно и распределено по некоторому закону. Продвижение заявок из позиции в позицию в переходах "индивидуально" — переход может изымать, выдавать и задерживать фиксированное число меток, соответствующее 1 заявке, то есть по функциям инцидентности F и H.

Все варианты соответствуют вычислению последовательных состояний системы через некоторые интервалы времени, что соответствует известному “принципу At“ в моделировании [9]. Заметим, что другие известные принципы организации функционирования имитационных моделей, например “последовательной проводки заявок” [9], мало пригодны для рассматриваемого класса моделей реального времени с несколькими видами параллелизма и большим числом одновременно обслуживаемых заявок.

Первый вариант рассматривает кванты времени, как фиксированные величины, не зависящие от интенсивности поступающей нагрузки, что безусловно удобно с точки зрения времени моделирования. С другой стороны нетрудно заметить, что такой подход значительно усложняет логику работы перехода, требует дополнительных модификаций сети (раскрашивания меток, переменной кратности инцидентных дуг, “вложенных” тактов работы и т.п.), что делает его менее предпочтительным.

Второй и третий варианты рассматривают кванты времени, как величины, в течении которых вероятность поступления двух и более меток пренебрежимо мала, то есть зависимые от интенсивности нагрузки, что ведет к непостоянству времени моделирования. Очевидно, что в этих вариантах время как бы "растянуто", процессы рассматриваются более подробно, продолжительность исследования больше. Отличие вариантов в логике работы сети в тех случаях, когда случается одновременное (в одном такте) поступление нескольких вторичных заявок одного типа, — то есть нескольких меток в одну позицию. Второй вариант предполагает одновременное изъятие, продвижение и выдачу всех меток , а третий — последовательное по тактам: сначала одна, затем вторая и т.д. Для детерминированной подсети алгоритма более естественным и предпочтительным представляется третий вариант, который также хорошо согласуется с принципом поступления меток. Для стохастической подсети внешнего окружения в каждом

такте производится "розыгрыш" дальнейших путей следования меток из позиций, поэтому по третьему варианту необходимо выборочно, по каждой метке, запоминать "выпавший путь" на последующие такты, что нежелательно. Практические исследования показывают, что для реальных значений интенсивности нагрузки УК малой и средней емкости вероятность поступления в ЖГСМ однотипных вторичных заявок в одном такте невелика. Поэтому компромиссным решением представляется выбор третьего варианта, с дополнением логики работы ЖГСМ *управляющей функцией,* которая производит в каждом такте предварительное изъятие меток из позиции в соответствии с "розыгрышем" пути, если число меток в позиции ЖГСМ больше одной.

Заметим так же, что с целью упрощения модели можно не учитывать некоторые сигналы УК во внешнюю среду (п.2.1), если известно, что минимальное время реакции абонента превышает максимально возможную задержку выдачи сигнала из УК (например включение-выключение выдержки времени и т.п.). С точки зрения формального описания они являются обязательными, однако при имитационном моделировании действие их ничтожно или отсутствует вообще. Кроме того, модель может намеренно не рассматривать выдачу некоторых сигналов, чтобы исследовать поведение системы в специальных условиях (например при дисциплине обслуживания с ожиданием, для определение среднего времени задержки - не выдавать СЗ и т.д.)

Описанные основные принципы построения модели на основе СП определяют состав исходных данных, необходимых для моделирования.

Предлагаемое выше расширение СП для моделирования УК можно задать формально, как набор множеств NE:

$$N_E = \{P, D, F, H, Mo, т, T, Pr, Go, G, p_G\}, \qquad (4.30)$$

где P— множество всех вершин позиций,

D—множество всех вершин переходов,

F— прямая функция инцидентности,

H— обратная функция инцидентности,

Mo— начальная разметка сети,

т— временная база сети,

T— вектор временных задержек переходов,

Pr— вектор приоритетов переходов,

Go— вершина-позиция генератора меток (GuGo) cP,

G— множество вершин-позиций специального типа — ЖГСМ,

p_G— множество векторов распределения вероятностей по выходам ЖГСМ G):

$$Pвп \sim \{PGn > PGm' ---> PGk\}\ '$$

где n, m, к — номера позиций ЖГСМ,

$Pвпт\ к$— вектора распределения вероятностей по выходам соответствующих позиций $G_{n\ m\ к}$.

Задание конкретной модели состоит в конкретизации (однозначном определении) множеств из набора Nᴇ. Очевидно, что описанная выше методика сетевого описания УК позволяет однозначно определить множества P, D, Go, G. Функции инцидентности F и H определяются частично, без учета кратности дуг. Таким образом, после выполнения сетевого описания, необходимо определить множества Mo, т, T, Pr, p_G и доопределить кратности дуг по функциям инцидентности F и H.

Начальная разметка Mo отражает:

- число вызовов, находящихся на различных стадиях обслуживания (подсеть алгоритма) в момент начала функционирования сети;
- состояние окружения УК — сколько и каких заявок должно поступить в ответ на поступившие ранее внешние сигналы УК к моменту начала функционирования сети;
- состояние ресурсов УУ УК — сколько каких ресурсов свободно, задействовано, когда должны освободиться — на момент начала функционирования сети .

Концепция присвоения временных задержек переходам УК предполагает, что метки поглощаются переходами, в случае их готовности к срабатыванию, на время задержки данного перехода. Тогда в течение этого времени метка не находится ни в одной из позиций сети. Данное обстоятельство исключает возможность адекватного задания некоторого промежуточного состояния УК начальной разметкой моделирующей сети Mo. Поэтому будем считать корректной Mo, в которой:

1. Разметка позиции множества Go (в модели — Pi) равна единице (соответствует поступлению первого вызова в начальный момент времени Mo (Go) = 1).

2. Разметка позиций подсети ресурсов соответствует их исходному состоянию (ни один вызов пока не был принят на обслуживание).[1]
3. Разметка всех позиций подсети внешнего окружения, исключая Go, равна нулю.
4. Разметка всех позиций подсети алгоритма соответствует состоянию ожидания первого вызова (для дисциплин обслуживания с явными потерями и с ограниченным ожиданием — нулевая).
5. Разметка позиций подсети анализа — соответствует алгоритму определения искомой величины.

При выполнении данных требований, разметка Mo соответствует начальному включению УК во время ЧНН. В реальных условиях ЧНН наступает после того, как УК уже некоторое время работал, его ресурсы частично задействованы, на обслуживании находится некоторое количество вызовов. Такое состояние наступит в модели через некоторое время после начала функционирования. Назовем такое состояние установившимся (стационарным). Очевидно, можно считать, что его признаком является приблизительное равенство интенсивностей потока занятий и потока освобождений. Постоянный контроль интенсивности потока освобождений заметно замедлит работу моделирующей программы, поэтому ориентировочное время установления стационарного состояния можно единожды определить экспериментально.

Таким образом, конкретизация начальной разметки состоит в выборе ее по указанным правилам и определении времени установления стационарного состояния.

Физический смысл величины временной базы сети — интервал времени, соответствующий 1 такту работы сети. Чем меньше т, тем выше точность моделирования временных соотношений в модели, но и тем большее число тактов необходимо для моделирования некоторого отрезка времени работы оригинала, что увеличивает время моделирования. Ясно, что в случае пуассоновского потока, при большей интенсивности поступающих вызовов, т должна быть меньше, так как уменьшается средняя величина интервала между соседними вызовами. Концепция сетевой модели, как и распределение Пуассона, предполагает, что одновременное поступление двух и более вызовов невозможно. Отсюда принцип выбора *x:* это интервал времени, в течение которого вероят-

[1] Определение конкретных значений разметки позиций подсети ресурса рассмотрено ниже.

ность поступления более 1 вызова не превышает некоторой разрешенной величины. Это условие можно выразить по другому: вероятность превышения интервала между соседними вызовами величины т не более заданной (допустимой):

$$p0>т)<p_{до},, \qquad (4.31),$$

где At — интервал между соседними вызовами,

Рдоп — допустимый предел превышения вероятности.

На рис.4.1 показан пример функции распределения At (экспоненциальный закон) и принцип выбора т.

Для случая пуассоновского распределения вызовов с параметром X имеем экспоненциальное распределение для At:

$$p = 1— e^{л‘}.$$

При t = т р = рдоп , тогда, выполнив подстановку и решив уравнение относительно т имеем:

$$m=\sim 1n(\backslash -p_{доп}) \blacksquare \qquad (4.32)$$

Л

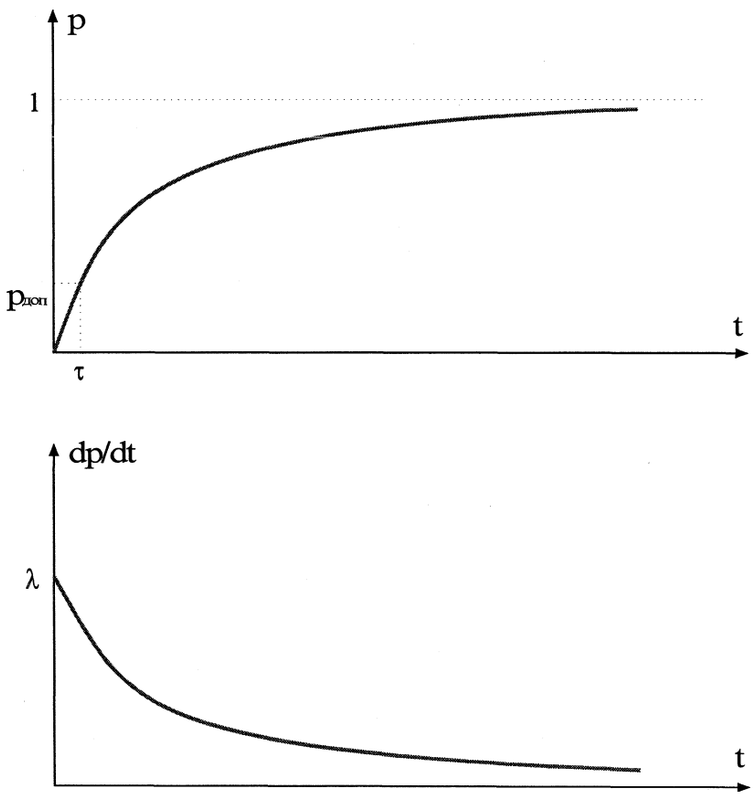


Рис.4.1. Пример закона распределения At (экспоненциальный)

Таким образом для определения т необходимо задаться допустимой вероятностью Рдоп поступления более 1 вызова за т и интенсивностью потока вызовов Х.

Заметим, что в общем случае поток поступающих вызовов может быть и не пуассоновским. В этом случае т определяется аналогично, но уже по другому закону распределения промежутков между вызовами.

Временные задержки переходов пересчитываются в такты *(t,)* делением:

$$ti = \sim \frac{T}{m},$$

с округлением по обычным правилам. Конкретные значения времен задержки назначаются в соответствии с длительностью протекания события, моделируемого данным переходом:

- длительности ожидания, прослушивания сигналов ОС и СЗ, разговоров, набора номера и т. п. определяются на основе анализа спектра занятий, статических данных существующих УК, данных хронометража и т. д. (подсеть внешнего окружения);
- длительности выполнения операций по обслуживанию заявок определяются на основе анализа временных соотношений при отработке частных алгоритмов (п.2.1.2) с учетом быстродействия элементной базы аппаратных средств, на которых предполагается реализация УУ УК (подсеть алгоритма);
- задержка переходов подсети ресурсов (при наличии таковых), отражающих динамику перераспределения ресурсов для выполнения тех или иных операций, можно считать нулевыми;
- задержки переходов подсети анализа — нулевые.

Установление отношений приоритетности для переходов подсети алгоритма должно быть таким, чтобы:

1. Обеспечивалась правильная логика смены стадий обслуживания по каждому вызову в соответствии с состоянием ресурсов (определяется системой обслуживания — с потерями, с ожиданием и т.д.).

2. Относительные приоритеты переходов, конкурирующих по ресурсным позициям должны соответствовать уровням приоритета по прерыванию отображаемых ими частных алгоритмов в соответствии с матрицей защиты от прерываний.

Как было указано в п.2.1, более естественным и наглядным способом отражения логики ресурсных отношений является использование ингибиторных дуг. Однако в большинстве реальных алгоритмов число таких дуг оказывается большим, что исключает указанное преимущество. Как показано в [6], для любой ингибиторной СП можно построить эквивалентную ей приоритетную СП. При этом последняя намного менее затенена дугами, легче читается и проверяется, что делает целесообразным выбор средства описания ресурсных отношений и УК в целом в пользу приоритетных СП. Ингибиторные СП можно использовать как промежуточное представление тогда, когда это удобно, с последующим переходом к приоритетным по соответствиям (рис.4.2). Заметим, что для небезопасных СП с произвольной кратностью дуг соответствия не являются полными, так как в случае M(pi)<F(di,pi) в ингибиторной СП оба перехода не готовы, а в приоритетной готов к срабатыванию c^. Однако, логика приоритетной СП как раз более точно отображает ресурсные отношения, так как отмечает не только полное отсутствие, но и недостаточность ресурса для обслуживания вызова на некоторой стадии.

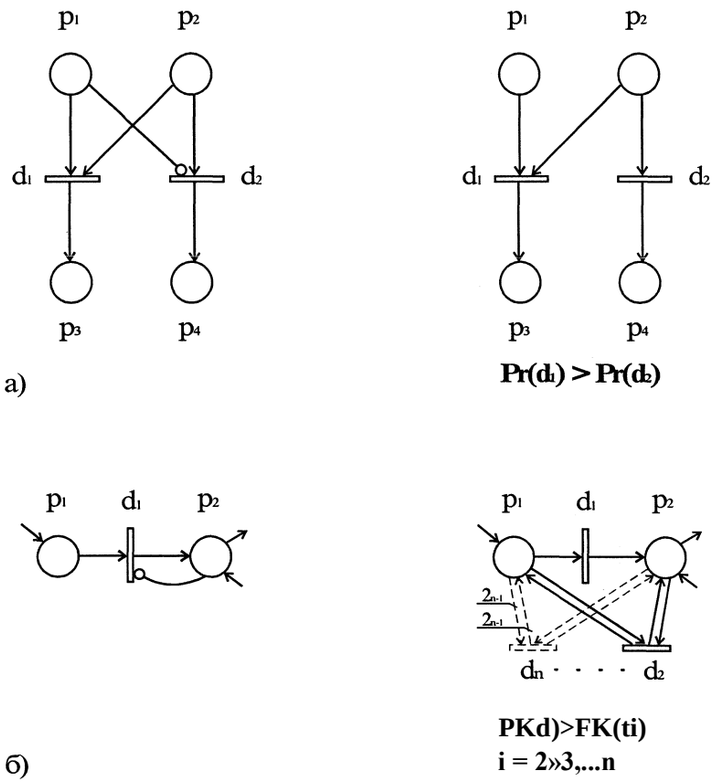


Рис.4.2. Соответствия ингибиторной и приоритетной сетей

Фрагмент СП на рис.4.2.б отражает не ресурсные отношения, а блокировку, что не характерно для описания УК и встречается очень редко. Поэтому, несмотря на большую

графическую избыточность приоритетного варианта, в необходимых случаях он может использоваться, что сохраняет общие правила функционирования сети.

Приоритеты переходов подсети анализа выбираются в каждом конкретном случае так, чтобы обеспечить правильность формирования искомых величин (см. п.4.2.2). Приоритеты конкурирующих переходов в подсети внешнего окружения являются случайными величинами, формируемыми ЖГСМ (п. 2.3). Приоритеты переходов, свободных от конкуренции, безразличны и могут не назначаться.

Определение векторов распределения вероятностей по выходам ЖГСМ, как и временные задержки их выходных переходов, производится на основе спектра занятий, статистики, хронометража. При этом закон распределения аппроксимируется ступенчатой кривой или гистограммой, выбираются характерные интервалы значений отрезков времени и определяется вероятность попадания в каждый из них. Среднее значение интервала дает задержку соответствующего перехода, а вероятность попадания в него — компоненту вектора распределения. Ясно, что необходимым условием является

$$\sum_{i=i}^{n} 2>,=ь \qquad (4.33)$$

где pi — i-я компонента вектора распределения;

n — число выбранных интервалов.

Примеры, поясняющие определения задержек и векторов распределения ЖГСМ, показаны на рис.4.3. На рис.4.3.а показан вариант использования функции вероятности. Компонентами вектора являются отрезки ОА, АВ, ВС, СД. На рис.4.3.б показано использование плотности вероятности. Компонентами вектора являются площади прямоугольников с основанием, равным соответствующему интервалу времени и высотой — усредненной плотностью вероятности по этому интервалу. Во втором случае необходимо нормировать величины pi —P5 для выполнения условия (4.33).

Определение конкретных значений начальной разметки ресурсных позиций производится в зависимости от конкретного семантического содержания каждой позиции. Как было показано в п.2.3, ресурсы УУ УК можно разделить на 3 группы:

1. Ресурсы, характеризуемые числом обслуживающих устройств (ПНН, ПАС и т. п.).
2. Массивы в ОЗУ различного назначения (МОВ, МЗО и т. п.).

3. Реальное быстродействие процессора (микроЭВМ), определяющее число заявок разного типа, для которых может быть реализовано одновременное (многозадачное с разделением времени) обслуживание.

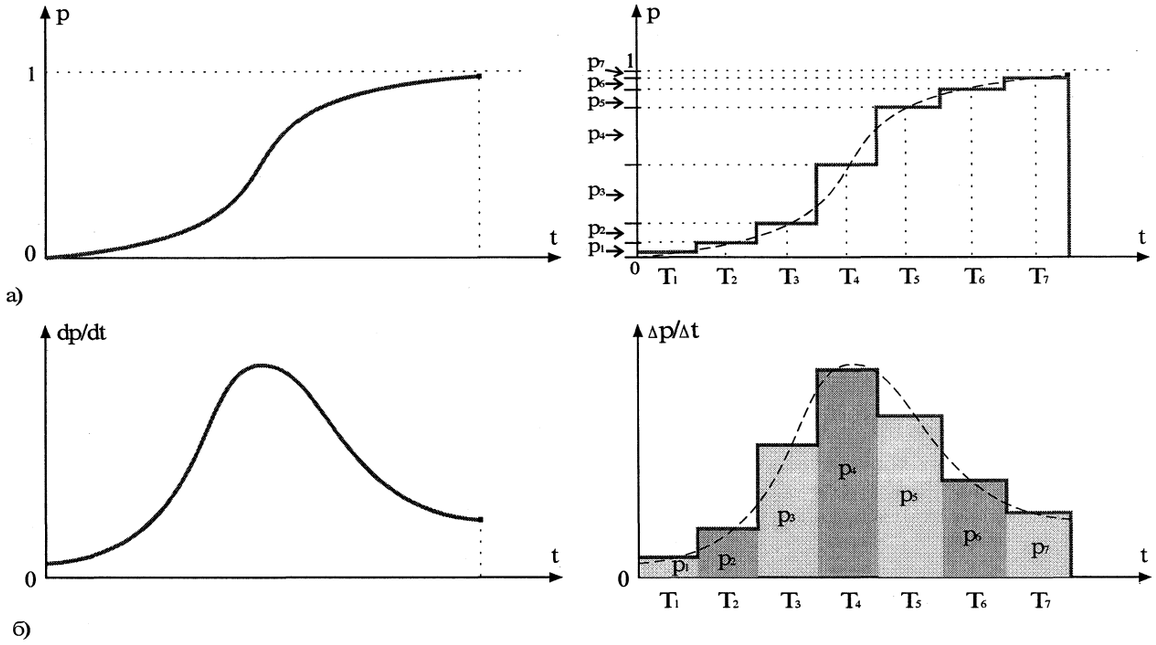


Рис.4.3. К заданию законов распределения ЖГСЧ

Определение начальной разметки и кратностей дуг для ресурсов первых двух типов описано в п.2.3. Способ определения времен задержки переходов и начальной разметки ресурсов третьего типа зависит от типа используемых ЭУС и ОС. В простейшем случае циклической организации обслуживания, необходим анализ цикла работы процессора (Тц) с целью определения части Тц, отводимого под выполнение программ различных частных алгоритмов. При известном среднем времени выполнения совокупности операций i-го частного алгоритма, временную задержку t_t соответствующего ему перехода, можно определить по формуле

$$Л\ П,\ Тц, \tag{4.34}$$

где *nt* — число циклов процессора, необходимых для отработки i-го частного алгоритма:

$$n_i = \left[\frac{k_i Tц}{T_i}\right], \tag{4.35}$$

где kj — доля Тц, отводимая для выполнения i-го частного алгоритма;

Tі — среднее время выполнения совокупности операций i-го частного алгоритма (п. 2.1.2);

[...] — знак округления до большего целого.

Часть времени Тц. В течение которого выполняются алгоритмы одного класса с данным (t_{ra}) может быть определена, как сумма всех отрезков времени, в течение которых процессорное время предоставлено данным алгоритмам.

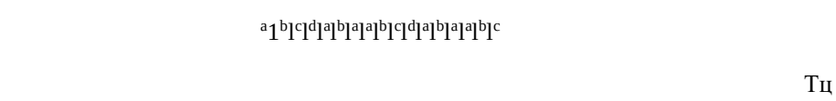


Рис.4.4. Пример структуры цикла работы процессора

Для класса "a" в примере на рис.4.4 это время будет равно сумме отрезков, обозначенных "a", (1кл = St_a). Доля первого алгоритма определяется вероятностью его выполнения pi, среди алгоритмов одного класса.

$$\frac{'Pi}{Tц} \qquad (4.36)$$

Тогда, подставив (4.35) и (4.36) в (4.34), получим:

$$. = Tц^{pt} \qquad (4.37)$$

В более сложных ОС реального времени — с многоуровневыми прерываниями и контролем допустимого времени выполнения — распределение периодов отработки некоторого частного алгоритма по одной заявке имеет сложный, стохастический характер. Поэтому, в зависимости от целей моделирования, допустимой точности соответствия временных задержек переходов фактическому среднему времени выполнения алгоритма, можно поступить по одному из вариантов:

- назначить задержки равными допустимому времени выполнения алгоритма:

  6' $_{Рдоп\ i\ >}$

- на основе оценки типа смесей Гиббсона (GAMM, POWN и др.) или бенчмарковских программ определить среднее время выполнения соответствующего набора операций в условиях, отражающих реальную работу УУ;

- составить имитационную модель отработки алгоритма (см. п.2.1.2), учитывающую фактор случайного числа прерываний со случайной длительностью, и произвести испытания модели, на основе чего и определится t_1.

Определение начальной разметки и кратности инцидентностей для ресурсных позиций типа производительности процессора необходимо выполнять совместно. Точное определение этих величин затруднено многообразием принципов работы операционных систем и тем фактом, что в большинстве многозадачных ОС реального времени характер распределения ресурсов, длительности выполнения частных алгоритмов имеют сложный, непостоянный характер и не могут быть отражены постоянными числами. Попытка адекватного описания ОС значительно усложнит и без того объемную модель. Приблизительную оценку начальной разметки и кратностей инциденций для таких позиций можно произвести в предположении, что время отработки частного алгоритма в течении ЧНН не зависит от нагрузки и постоянно, в несколько этапов:

1 . Определить, какое максимальное число заявок может обслужить процессор, при условии, что выполняется только алгоритм, описываемый одним из выходных переходов. Назовем это число емкостью по данному переходу. Очевидно такая емкость определится соотношением:

$$\frac{\cdot}{{}^{1}\,i} \qquad (4.38)$$

где Zj — емкость ресурса по i-му прямо инцидентному переходу;

T_{AI} — средняя длительность выполнения i-го алгоритма в многозадачной ОС реального Времени, $T_{AI} < T_{доп\ i}$,

где $T_{доп\ I}$ — допустимое время выполнения i-го алгоритма;

T_I — сумма длительностей операций i-го алгоритма (см. п. 2.1.2);

Результатом определения Zi по каждому из выходных переходов является множество емкостей

где ш — число переходов прямо инцидентных ресурсной позиции процессора.

2 . Поскольку величина производительности процессора, отвлекаемая алгоритмом по данному переходу тем больше, чем меньше его емкость по тому же переходу, можно

нормировать эту производительность так, чтобы алгоритм наибольшей емкости отвлекал единицу производительности. Это соответствует первоначальному назначению разметки данной ресурсной позиции, равной максимальному элементу множества z:

$$M'(Pu) = max\ \{z_h\ z_2,\ldots,\ z_m\}, \qquad (4.39)$$

где M'(Рп) — первоначальная разметка ресурсной позиции процессора.

3 .При первоначальной разметке, кратности прямых инциденций к другим переходам определятся в соответствии с пропорцией емкостей:

$$F'(P_n\ ,\ d.) = MJInL. \qquad (4.40)$$
$$^{z}i$$

4 .Поскольку кратности инциденций могут принимать только натуральные значения, необходимо произвести их выравнивание так, чтобы сохранилась пропорция (4.40). Это можно выполнить, например, по следующему алгоритму:

- округление кратностей F' до n-го знака после десятичной запятой;
- умножение на 10^n:

$$F''=10^nF^{>}\text{-}, \qquad (4.41)$$

- нахождение наибольшего общего делителя *I* для кратностей F'':

$$l = Hod\{F(P_n\text{-}dM, \qquad (4.42)$$

где *i* - номер прямо инцидентного перехода,

нод - операция нахождения наибольшего общего делителя;

- деление кратностей F'' на найденный делитель *I* — получаем искомую кратность F:

$$F = \frac{F''}{I}; \qquad (4.43)$$

- коррекция значения разметки ресурсной позиции для учета преобразований кратностей инциденций:

$$M(P_n) = 10^n\wedge(^{Pn}) . \qquad (4.44)$$

Заметим, что число значащих цифр определяется требуемой точностью выдержки соотношения между разметкой по разным ресурсам.

4.2.2. Определение показателей функционирования по сетевой модели

Рассмотренные принципы построения сетевой модели позволяют отразить свойства внешней среды, временные и ресурсные характеристики моделируемого УК. Однако главной целью моделирования является получение прогноза на количественные показатели качества обслуживания. К наиболее значимым показателям следует отнести:

1. В системе с явными потерями:
   1.1. вероятность потери вызова:

$$P_{пот} = \frac{c_{пот}}{c_0} \qquad (4.45)$$

где Спот — средняя интенсивность потока вызовов, не обслуженных УК в некотором промежутке времени;

со — средняя интенсивность потока поступивших на обслуживание вызовов в том же промежутке времени.

   1.2. вероятность потери по времени:

$$п' = \frac{}{t_{пот}}, \qquad (4.46)$$

где ton — "опасное время", то есть время, в течение которого УК находится в состоянии исчерпанности тех ресурсов, которые необходимы для взятия очередного вызова на обслуживание,

t— общее Время, В течение которого определялось ton.

   1.3. Вероятность потерь по нагрузке

$$У = \frac{c_{пот}}{c_п}; \qquad (4.47)$$

где Спот — интенсивность потерянной нагрузки,

Сп — интенсивность потенциальной нагрузки.

2. В системе с ожиданием:
   2.1. Вероятность ожидания по вызовам:

$$V = \frac{}{N_{ож}}, \qquad (4.48)$$

где No» — число вызовов, при обслуживании допущено ожидание;

N — общее число вызовов.

Величину Рож иногда определяют по времени

$$P_{ож} = \frac{\sum_{j=1}^{n} t_{ожj}}{t} \qquad (4.49)$$

где п — число стадий обслуживания;

$t_{ожI}$ — длительность ожидания любого количества вызовов на -й стадии обслуживания;

t — время, в течение которого производились фиксации $I_{ожI}$.

2.2. Вероятность ожидания сверх допустимого времени по вызовам:

$$P_{ОЖ} = \frac{N_{ожс}}{N}, \qquad (4.50)$$

где Иожсі — число вызовов, ожидание обслуживания которыми превысило допустимую величину ІД.

2.3. Среднее время ожидания

- по поступившим вызовам:

$$t_{ср} = \frac{\sum_{j}\sum_{i} t_{ожу}}{N_{ож}} \qquad (4.51)$$

где Сж у — время ожидания і-м вызовом на j-й стадии обслуживания;

N — число поступивших вызовов;

- по задержанным вызовам:

$$t_{ср} = \frac{\sum_{j}\sum_{i} t_{ожу}}{N_{з}}, \qquad (4.52)$$

где $N_{з}$і — число задержанных вызовов на і- й стадии обслуживания.

2.4. Средняя длина очереди заявок на некоторой стадии обслуживания:

$$S = \sum_{t} \frac{\tau}{t}, \qquad (4.53)$$

где Si — начальная длина очереди, Si=l;

S_m — максимальная длина очереди.

і — целочисленный индекс одной из возможных градаций длин очередей,

0< 1 < т;

ti — суммарное время существования очереди длиной Si;

t — общее время наблюдения.

2.5. Вероятность превышения допустимой длины очереди на данной стадии обслуживания:

$$p_{n.o} = \frac{\sum_i t_{n.oi}}{t}, \qquad (4.54)$$

где t_n.o — время, в течение которого очередь i-го вызова была более допустимой.

2.6. Вероятность превышения допустимого времени ожидания сигналов “ответ станции” и “контроль посылки вызова” (времени задержки при коммутации).

2.7. В некоторых случаях дисциплина обслуживания устанавливается таким образом, что при отсутствии необходимого ресурса в момент запроса, вызов становится на ожидание с заранее установленным временем, после чего производится повторная попытка. В общем случае число попыток может быть разным, но конечным. Тогда обслуживание может быть охарактеризовано средним числом попыток обслуживания *ïï:*

$$\frac{\quad}{N}, \qquad (4.55)$$

где N — число вызовов, принятых к обслуживанию,

щ — число попыток по обслуживанию на данной стадии i-го вызова.

Описание такой процедуры СП внешне не отличается от описания дисциплины с ограниченным ожиданием, разница — во временных задержках переходов (рис.4.6).

3. Комплексный показатель, часто используемый для сравнения УК — пропускная способность, под которой понимается максимальная интенсивность обслуженной нагрузки, при которой обеспечивается нахождение выбранной группы показателей в заданных пределах. Кроме того представляет интерес оценка предельных значений разметки некоторых ресурсных позиций, что позволило бы судить о степени избыточности запаса данного ресурса. Непосредственную оценку перечисленных величин по сетевой модели, состоящей из 3-х подсетей сделать сложно.

Одним из возможных подходов к решению данной проблемы является построение дерева достижимых разметок сетевой модели и анализ его на содержание некоторой разметки или группы разметок. Несмотря на полноту, такое решение чрезвычайно избыточно. Поэтому представляется целесообразным другой подход, основная идея кото-

рого — наложение на сетевую модель еще одной — вычисляющей подсети, в позициях которой в процессе моделирования формировались величины, по которым можно непосредственно получить интересующие исследователя качественные показатели.

Заметим, что в большинстве реализаций УК система обслуживается вызовов является комбинированной: на одних стадиях явные потери, на других — ожидание, на третьих — ограниченное ожидание, повторные попытки. Поэтому один и тот же УК можно охарактеризовать показателями, присущими разным дисциплинам обслуживания вызовов. При этом подсеть алгоритма должна быть дополнена элементами, отражающими ту или иную дисциплину, как частный алгоритм, если это необходимо.

Элементы вычисляющей подсети в общей имитационной модели должны выполнять ту же роль, что и счетчики (занятий, отказов и т.п.) в реальных УК. Отсюда основной принцип построения подсети — установка накапливающих позиций, обратно инцидентно связанных со всеми теми переходами, которые моделируют интересующее событие. Если такие переходы уже имеются в сети, то достаточно соединить их обратно инцидентными дугами с позицией-счетчиком. Таким способом можно, например, получить общее количество поступивших No, потенциальных N_n и потерянных ВЫЗОВОВ NnoT для определения величин интенсивностей Спот и Со в формулах (4.45) и (4.47):

$$\frac{N}{t}, \qquad \frac{N}{t}, \qquad \frac{N}{t},$$

где t — время моделирования.

Пример, поясняющий подключение позиций-счетчиков показан на рис.4.5.

Для определения различных промежутков времени, входящих в формулы (4.46), (4.49), (4.51) 4- (4.53) непосредственное подключение накапливающих позиций обычно невозможно, так как в системе с ожиданием нет переходов, моделирующих ожидание, которое либо связывается с нахождением метки в некоторой позиции (несрабатыванием ее выходных переходов), либо с перемещением ее из позиции в позицию в течение некоторого числа тактов работы сети. В первом случае, необходимо создать в вычислительной подсети переходы, моделирующие нахождение метки (меток) в интересующей позиции в течение одного такта модельного времени, а во втором — подсчитать число срабатываний соответствующих переходов. Заметим, что дисциплина обслуживания с ожиданием в чистом виде практически редко реализуется из-за резкого снижения живучести

связи в аварийных и предельных ситуациях, ограниченности массивов памяти и т.д. Поэтому практический интерес представляет подсчет времени ожидания при ограничении ожидания как по времени, так и по длине очереди.

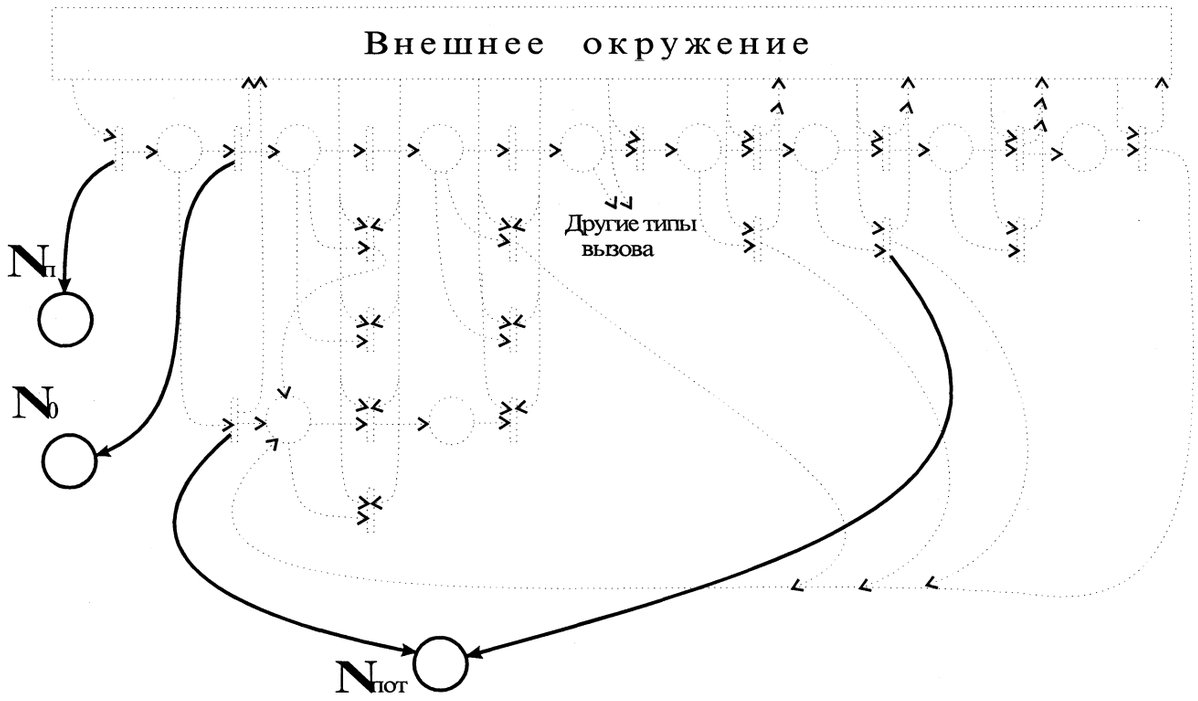


Рис.4.5. Включение позиций-счетчиков

Примеры, поясняющие подсчет временных параметров при различных дисциплинах обслуживания представлены на рис.4.6, 4.7. В случае дисциплины с ограничением времени ожидания в основном алгоритме имеются переходы, моделирующие ожидание в течение принятой градации времени (на рис.4.6 — сЬ-кЦ). Для подсчета суммарного времени ожидания (4.51), (4.52) позиция-счетчик соединяется обратными дугами с этими переходами. Кратность дуг назначается равной числу тактов временной задержки связанного перехода. Таким образом в позиции рю будет сформирована величина *1₀жу* , входящая в (4.51),(4.52). Величина Ыож (4.48) — это число срабатываний d_3. Для получения No® из (4.50) счетчик соединяется не со всеми переходами, а только с тем, который моделирует задержку сверх допустимого времени (на рис.4.6, например — d_5). Аналогично определяется величина £гц (4.55).

При необходимости получения закона распределения $1_{0ж}$у, к тем же переходам подключаются отдельные позиции, по разметкам которых после испытаний модели

строится гистограмма, ряд (многоугольник) распределения, которые и определяют закон распределения $1_{ожц}$ (см. п.4.2.1).

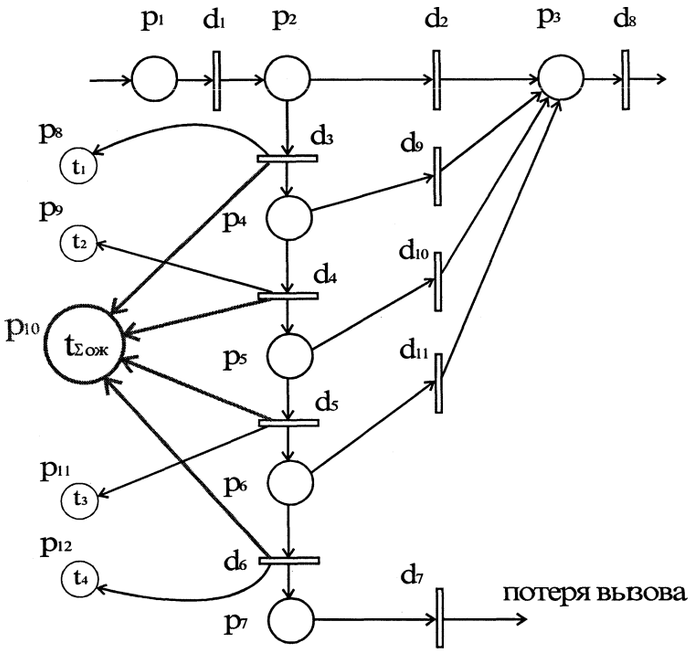
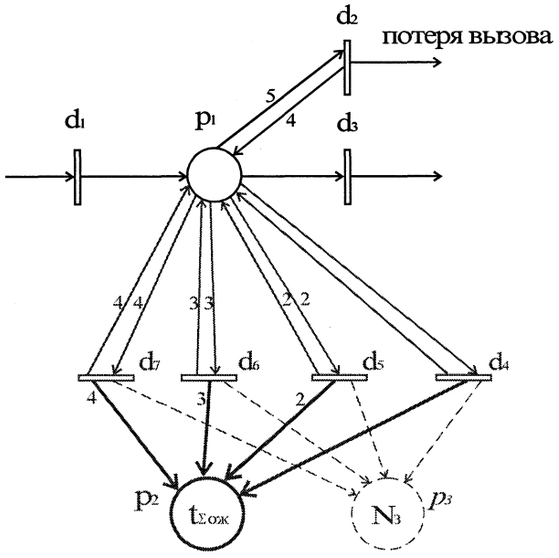


Рис.4.6. Дисциплина с ограниченным ожиданием по времени

В случае дисциплины с ограничением очередей и ограниченным временем ожидания (рис.4.7) обычно имеется переход, моделирующий потерю вызовов при переполнении позиций, то есть, когда ее разметка превышает допустимую величину (на рис.4.7 — сіг). Поскольку переходы, моделирующие временную задержку отсутствуют, их необходимо создать в вычисляющей подсети (на рис.4.7 — 64—d_7). Каждый из этих переходов моделирует задержку на один такт числа меток, равного кратности инцидентных ему дуг. При выборе приоритетов так, как это показано на рисунке за один такт модели сработает один из сЦ—d_7. Таким образом в рг число меток будет соответствовать суммарному времени ожидания всех вызовов на данной стадии обслуживания.

Как и в предыдущем случае, подключение к переходам отдельного счетчика единичными инцидентными дугами позволяет получить закон распределения, но уже не времени ожидания, а длин очередей S. Что касается величины S $S_t t_t$ (4.53), то, как видно из рисунка, она численно равна t_{OMi}. . Действительно, в каждом такте работы сети один из переходов сЦ—d_7 выдает в счетчик число меток, равное длине очереди, то есть S_t . За некоторый интервал времени каждый переход сработает t_t раз и выдает в счетчик St

ti меток, которые в сумме образуют величину Поэтому для вычислений по формуле (4.53) в числитель выражения можно подставить /ож•;7 , то есть разметку счетчика р2. Для определения величины N3i (4.52) можно использовать счетчик рз, в котором 1 метка будет добавляться всякий раз, когда имела место задержка вызова.



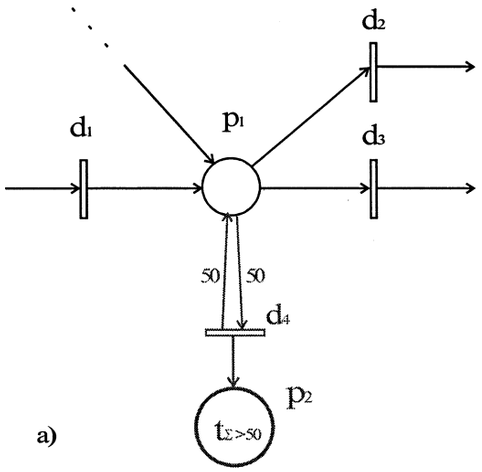
Pr(ds) > Pr(cЬ) > Pr(<1) > Pr(<±) > Pr(cL)

Рис.4.7. Дисциплина с ожиданием и ограниченными очередями

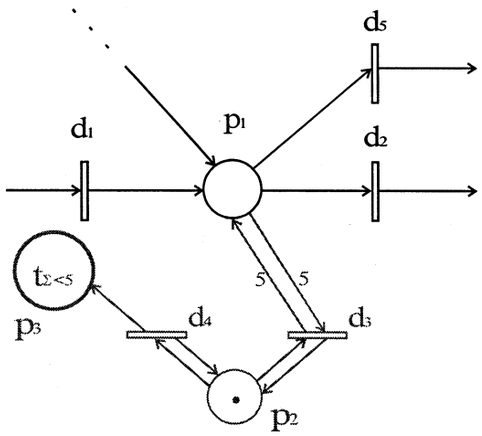
В ряде случаев представляет интерес вопрос о нахождении разметки интересующей позиции в некоторых заданных пределах в процессе функционирования сети, и в течение какого времени она выходила за (достигала) предельные значения. Примером МОЖеТ СЛуЖИТЬ Определение ВеЛИЧИНЫ ton (4.46), физический смысл которой можно трактовать как время (число тактов), при котором наблюдалась исчерпанность любого из ресурсов, необходимых для взятия очередного вызова на обслуживание. В терминах СП это означает, что в некоторой позиции необходимо зафиксировать метку в каждом такте, в котором разметка любой из заданного подмножества позиций была не более некоторой величины. Таким образом необходимо фиксировать ситуацию уменьшения разметки до некоторой величины.

Другой задачей является контроль превышения очередью заявок некоторой величины, то есть определение величины ton п (4.54). В этом случае необходимо фиксировать факт достижения разметкой заданной величины снизу. Примеры, иллюстрирующие возможный подход к решению задачи, показаны на рис.4.8. Контроль превышения за-

данной величины можно получить соответствующим выбором кратности дуг к переходу, отмечающему это событие (рис.4.8.а). Контроль снижения разметки можно получить, рассматривая это событие как альтернативу превышения разметкой порога снижения. На рис.4.8.б. переход da отмечает последнее событие. При условии pr(d4) < pr(d_3), срабатывание сЦ произойдет только в случае, если не сможет сработать d_3, что в свою очередь возможно при недостаточном количестве меток в pi. Заметим, что помимо описанных возможны и другие методы контроля величины текущей разметки в позиции.



а)



Pr(Щ) < Pr(дз) < min{Pr(F(pi))} — здесь F(pi) = {d>, ds}

б)

Рис.4.8. Контроль допустимой величины разметки: а) на превышение максимальной, б) на занижение ниже минимальной.

Таким образом при использовании описанных приемов можно получить основные величины, необходимые для подсчета качественных показателей функционирова-

ния УК по формулам (4.45)^-(4.55). Пропускную способность можно определить подбором интенсивности поступающих вызовов при условии контроля этих показателей.

Резюмируем методику определения показателей функционирования УК по сетевой модели в виде следующей последовательности действий:

1. Задаться перечнем интересующих показателей, представить их в виде формул (4.45)4-(4.55) или им подобных, по которым сформировать список искомых величин, представленных целыми положительными числами (число заявок, такты времени).

2. Определить, имеется ли в сетевой модели переход, непосредственно моделирующий событие, число реализаций которого и составляет искомую величину. При наличии такого перехода образовать в подсети анализа позицию-счетчик, которую соединить с найденным переходом обратной инцидентной дугой (рис.4.5). Для определения суммарного показателя счетчик соединяется со всеми такими переходами (рис.4.5, 4.6).

3. При отсутствии указанных переходов непосредственно в сетевой модели УК,— создать их в подсети анализа. При этом факт достижения некоторой контрольной разметки снизу ($M>M_K$) будет отмечен срабатыванием перехода, если он прямо и обратно инцидентен контролируемой позиции с кратностью, равной контрольной разметке (рис.4.7, 4.8.а). Достижение контрольной разметки сверху ($M<M_K$) можно отметить, как срабатывание альтернативного перехода по отношению к переходу отмечающему ($M>M_K$). Для этого использовать дополнительную служебную позицию (рис.4.8.б).

4. Назначить приоритеты переходов подсети анализа так, чтобы:
   - срабатывание переходов подсети анализа происходили после того, как сработают переходы сетевой модели УК, инцидентные контролируемым позициям;
   - соотношение приоритетов переходов подсети анализа обеспечивало правильность формирования разметки счетчиков.

5. Организовать функционирование полученной таким образом общей модели (включающей подсеть анализа) в течение заданного отрезка модельного времени.

6. Произвести расчет показателей функционирования по формулам (4.45)4-(4.55), используя разметку счетчиков с учетом времени задержек переходов.

такого перехода образовать в подсети анализа позицию-счетчик, которую соединить с найденным переходом обратной инцидентной дугой (рис.4.5). Для определения суммарного показателя счетчик соединяется со всеми такими переходами (рис.4.5,4.6).

3. При отсутствии указанных переходов непосредственно в сетевой модели УК,— создать их в подсети анализа. При этом факт достижения некоторой контрольной разметки снизу ($M>M_K$) будет отмечен срабатыванием перехода, если он прямо и обратно инцидентен контролируемой позиции с кратностью, равной контрольной разметке (рис.4.7, 4.8.а). Достижение контрольной разметки сверху ($M<M_K$) можно отметить, как срабатывание альтернативного перехода по отношению к переходу отмечающему ($M>M_K$). Для этого использовать дополнительную служебную позицию (рис.4.8.б).

4. Назначить приоритеты переходов подсети анализа так, чтобы:

- срабатывание переходов подсети анализа происходили после того, как сработают переходы сетевой модели УК, инцидентные контролируемым позициям;
- соотношение приоритетов переходов подсети анализа обеспечивало правильность формирования разметки счетчиков.

5. Организовать функционирование полученной таким образом общей модели (включающей подсеть анализа) в течение заданного отрезка модельного времени.

6. Произвести расчет показателей функционирования по формулам (4.45)-Н4.55), используя разметку счетчиков с учетом времени задержек переходов.

4.3 Обобщение метода моделирования узлов коммутации

Круг задач, для решения которых можно использовать методику моделирования узлов коммутации, предложенную в данной работе, рассмотрен в разделе. 1. Все они в той или иной степени связаны с инженерной деятельностью специалистов по технике средств распределения информации. В этом подразделе подводится итог данной разработки в виде общего описания последовательности действий по построению и испытанию описанных имитационных моделей на основе концепции, изложенной в п.4.2.1.

Этап I. Сетевое описание общего алгоритма функционирования УК.

Исходные данные.

- SDL-диаграмма процесса обслуживания вызовов.

Описание'.

1. Из элементов SDL-диаграммы выделить множество *SR,* состоящее из вершин СОСТОЯНИЕ и РЕШЕНИЕ, исключая вершину исходного состояния “Ожидание вызова”, “Свободно” и т.п.

2. Каждому элементу из этого множества *SR* поставить в соответствие позицию СП ф.(2.3), которую изобразить в слое 0 чертежа в графическом редакторе[1].

3. Установить между элементами множества *SR* отношения *предшествования и следования состояний* ф.(2.1) и (2.2), пометить *соседние* элементы. Заметим, что на данном шаге элементы, не принадлежащие *SR,* просто игнорируются.

4. Выделить множества дуг, исходящих из вершин множества *SR.* Каждой из них поставить в соответствие переход СП, который изобразить в слое 1 чертежа.

5. По надписям в вершинах SDL выяснить, имеются ли среди полученных переходов такие, которые соответствуют дисциплине обслуживания с ожиданием. При наличии размножить их копированием. Число копий равно числу аппроксимирующих интервалов времени ожидания на данном этапе (например, при среднем времени ожидания сигнала ОС t_{ox}-*3с* и интервале аппроксимации *At =1с,* число копий t_{oy}*JAt* =3).

6. Каждый полученный переход СП соединить обратной инцидентной дугой с той единственной позицией, которая отвечает вершине *SR, соседней* по отношению к образу

[1] Из известных автору графических сред в данное время наилучшим образом подходит AutoCAD R14 [80].

его входной позиции в направлении соответствующей дуги SDL. Присвоить дугам слой О чертежа.

7. Каждую позицию СП соединить прямой инцидентной дугой со всеми переходами, отвечающими одной из исходящих дуг соответствующей вершины ***SR.*** Присвоить дугам слой 0 чертежа.

8. Установить слой 2 чертежа. Образовать макропозицию внешнего окружения (ВО). Определить число вершин ВЫХОД — ***О,*** лежащих на пути между каждой парой соседних вершин из ***SR.*** Соединить переход, разделяющий их образы в СП, установленным числом ***О*** обратных инцидентных дуг с макропозицией ВО.

9. Определить число вершин ВХОД — ***I,*** лежащих на пути между каждой парой соседних вершин из ***SR.*** Аналогично соединить соответствующие переходы числом ***I*** прямых инцидентных дуг с макропозицией ВО. Заметим, что на этапе I, кратные дуги должны изображаться раздельно.

10. Пометить инцидентности к ВО типом входного (выходного) сигнала.

11. Придать чертежу максимальную наглядность компоновкой, пронумеровать позиции, начиная с 2, и переходы, начиная с 1 произвольным образом[1] .

Результат этапа I. Граф СП подсети алгоритма: позиции и инцидентные дуги — слой 0, переходы — слой 1, макропозиция ВО и инцидентности к ней — слой 2 чертежа.

Этап II. *Сетевое описание внешнего окружения.*

Исходные данные:

- SDL-диаграмма процесса обслуживания вызовов;
- граф СП подсети алгоритма, полученный на этапе I.

Описание:

1. По SDL-диаграмме составить граф смены сигналов, вершины которого — вторичные заявки из ВО, соответствующие ВХОДАМ и РЕШЕНИЯМ (рис.2.13).

2. Образовать первичную СП ВО. Для этого:

2.1. вершинам графа поставить в соответствие позиции СП, а пучкам дуг из одной вершины — переходы (рис.2.14);

2.2. каждую полученную позицию связать прямой инцидентной дугой с переходом, который образован по пучку исходящих из нее дуг;

[1] В прилагающейся программе использованы подписи позиций вида p7, p23... и переходов d2, diO....

2.3. каждому входящему сигналу при дугах графа поставить в соответствие позицию сети, которая связана обратной инцидентной дугой с подсетью алгоритма УК и прямой — с переходом сети, который был образован по пучку включающему эту дугу.

3. Определить наличие в сопрягаемой подсети алгоритма места локальных циклов и образовать в подсети ВО счетчики, с учетом нумерации УК (рис.2.18).

4. Преобразовать переходы сети первичной СП в ЖГСЧ (рис.2.15—2.17). Количество выходных переходов ЖГСЧ, моделирующего случайную временную задержку, определяется числом аппроксимирующих интервалов (рис.4.3). Позицию, соответствующую вершине "Занятие" графа объявить “автономным” ГСЧ.

5. Заменить в чертеже (слой 2) макропозицию ВО на чертеж подсети ВО. Произвести его компоновку.

6. Для передачи меток, моделирующих потоки заявок, в подсеть алгоритма — образовать сопрягающие позиции (на рис.2.19 показаны штриховой линией) и продлить ранее образованные дуги в слое 2 к этим позициям, и от них к соответствующим переходам подсети ВО.

7. Определить положение в СП подсети алгоритма переходов, срабатывание которых определяется вторичными заявками, к которым на первом этапе не были образованы обратные инцидентные дуги (соответствуют РЕШЕНИЯМ SDL) и образовать их в чертеже (слой 2).

8. При необходимости описания выдачи сигналов УК в ВО, изобразить соответствующие сопрягающие позиции с прямыми инцидентностями к переходам ВО, и обратными инцидентностями от переходов алгоритма, образованными на первом этапе (слой2). Иначе, эти дуги убрать.

9. Компоновка и нумерация элементов чертежа подсети ВО. Для удобства, позиции источника первичных заявок — “автономного” ГСМ, присваивается №1; нумерация выходных переходов каждого ЖГСЧ — упорядоченная по возрастанию.

10. ультат этапа II. Граф СП подсети ВО: все элементы — слой 2 чертежа. Подсеть ВО сопряжена с переходами подсети алгоритма.

Этап III. *Сетевое описание ресурсов подсистемы управления УК.*

Исходные данные:

- подробная структурная схема УК, на которой должны быть указаны:
  - все активные устройства (процессоры, микроЭВМ), исполняющие какие-либо коммутационные программы,
  - все пассивные устройства, используемые при установления и разрушения соединений (каналы СЛ, ПНН, датчики сигналов и др.),
  - схема включения абонентов в абонентские модули, закрепление обслуживающих приборов за нагрузочными группами;
- техническое описание процесса установления соединений, из которого можно установить на каких этапах захватывается тот или иной ресурс;
- данные о структуре массивов ОЗУ центрального процессора (для централизованных и иерархических ЭУС);
- данные о способе распределения памяти ОЗУ между массивами.

Описание:

1. На основе анализа структурной схемы УК определить состав ресурсных компонент, каждому из которых поставить в соответствии ресурсную позицию.

2. Создать текущий слой 3 чертежа, выключить слои 0 и 2. Изобразить ресурсные позиции. При динамическом распределении памяти ОЗУ описать его (рис.2.28, 2.30).

3. Проанализировать моменты участия каждого ресурса в алгоритме и отметить их прямыми и обратными инцидентными дугами к переходам подсети алгоритма.

4. Проанализировать закрепление ресурсных позиций за источниками заявок. В случаях неполнодоступного включения ресурсов произвести распараллеливание потоков меток по нагрузочным группам, используя один из описанных в подразделе 2.3 приемов:

4.1. создание параллельных участков ветвей алгоритма, соответствующих нагрузочным группам;

4.2. замена нескольких неполнодоступных источников нагрузки одним полнодоступным с пересчетом величины ресурса, обслуживающего полнодоступный пучок так, чтобы вероятность блокировки (ожидания) осталась неизменной (п.2.3, приложение Г.1);

4.3. использование управляющих функций того или иного вида, которые упрощают модель (см. п.2.3).

5. В соответствии с выбранным на шаге 4 приемом выполнить одно из следующих действий:

5.1. в случае 4.1 — включить все слои, образовать дополнительные слои по числу нагрузочных групп, скопировать распараллеливаемый участок полной сети в каждый новый слой с теми же координатами;

5.2. в случае 4.2 — пометка для пересчета ресурса на этапе V;

5.3. в случае 4.3 — дополнить чертеж подсети ресурсов (слой 3) специальными позициями (переходами) реализующими управляющую функцию.

6. Компоновка чертежа по слоям 1 + 3 + дополнительные (в случае шага 5.1). Нумерация элементов чертежа подсети ресурсов. Если были образованы новые позиции источника первичных заявок (шаг 5.1), то для удобства им присваиваются первые номера, с перенумерацией тех позиций, которым эти номера были присвоены ранее[1].

Результат этапа III. Граф СП подсети ресурсов: все элементы — слой 3 чертежа, в случае шага 5.1 — слой 3 + дополнительные. Подсеть ресурсов сопряжена с переходами подсети алгоритма.

Этап IV. Учет дополнительных факторов влияния. Задание элементов СП для определения показателей функционирования по сетевой модели УК.

Исходные данные:

- перечень дополнительных факторов влияния;
- перечень показателей функционирования УК подлежащих определению, и форма их представления;
- тип системы обслуживания на этапах установления соединений.

Описание:

1. Если постановка задачи моделирования требует учета дополнительных факторов влияния, создать новый слой и произвести сетевое описание таких факторов (п.2.4).

2. Определить состав потребных величин, которые должны сформироваться как разметки в позициях СП (п.4.2.2).

[1] процедура легко поддается автоматизации (приложение 3).

3. Определить тип системы обслуживания на тех этапах установления соединений, по которым вычисляются эти величины.

4. Произвести сетевое описание вычисления искомых величин (п.4.2.2, рис.4.5—4.8).

5. Изобразить полученные элементы на чертеже:

- при небольшом числе вычисляющих элементов СП их можно разместить на одноименных слоях с инцидентными им элементами СП (обычно слой позиций и инцидентностей подсети алгоритма — этап II, слой 0, или слой подсети ресурсов этап III, слой 3)[1];
- суммируемые величины формируются в одной общей позиции разных ветвей (слоев);
- если вычисляющие элементы СП заметно затеняют чертеж, их можно вынести в отдельный слой (слои) и предусмотреть соответствующее совмещение их со слоями инцидентных элементов.

Результат этапа IV. Граф полной сетевой модели дополнен элементами для вычисления показателей функционирования УК. Расположение их по слоям может быть различным (по построению).

Этап V. *Задание динамических параметров СП: вероятностей ЖГСЧ, временных задержек, приоритетов, начальной разметки.*

Исходные данные'.

- статистические данные о распределении времени формирования различных вторичных заявок в ВО или об условных вероятностях их поступления;
- данные о среднем времени выполнения частных алгоритмов активными элементами подсистемы управления УК;
- данные о максимальном числе одновременно обслуживаемых вызовов любым процессором при выполнении каждого из закрепленных частных алгоритмов (емкости ресурсов по переходам);
- вероятности блокировки в коммутационном поле при выбранном типе и параметрах потока вызовов;

1 Распараллеливание подсети алгоритма может повлечь распараллеливание вычисляющих элементов, в зависимости от типа вычисляемой величины.

- тип системы обслуживания на этапах установления соединений;
- матрица приоритетов выполнения частных алгоритмов.

Описание:

1. По статистическим данным о формировании различных вторичных заявок в ВО и вероятности блокировки в коммутационном поле назначить вероятности выходов ЖГСМ (файл[1] “gen” в примере приложения Г.1) и времена задержек их выходных переходов (файл “tim”).

2. По данным о среднем времени выполнения частных алгоритмов назначить временные задержки переходов подсети алгоритма. Временные задержки прочих переходов — нулевые (файл “tim”).

3. На основании данных о емкостях ресурсов по переходам (п.4.2.1, формулы (4.36) — (4.38)) выполнить приведение разметки к натуральным кратностям инцидентных дуг (формулы (4.39) — (4.44)). В результате определятся:

- величина начальной разметки ресурсных позиций, описывающих активные устройства (процессоры, микроЭВМ) — отразить в файле (в примере “Vr”);
- величина кратности инцидентных дуг — отразить на чертеже числом, пересекающим единственную инцидентную дугу в любом месте (слой кратности такой же, как и дуги);

4. Дополнить файл начальной разметки величинами ресурсов пассивных устройств, захват которых производится единственным переходом (ресурс не нуждается в приведении разметки), и, при необходимости, разметкой вспомогательных (вычисляющих) позиций (п.4.2.2).

5. В случае выполнения на этапе III:

5.1. шага 5.1 — перейти к 6.;

5.2. шага 5.2 — пересчитать объединенный ресурс (п.2.3, приложение Г.1);

5.3. шага 5.3 —задать управляющую функцию отдельным файлом “Uf”;

6. Назначить приоритеты переходов на основании анализа:

- системы обслуживания на каждом этапе установления соединения,
- матрицы приоритетов выполнения частных алгоритмов,

---

[1] в примере файл “Gen” создается программой обработки чертежа СП (приложение 2), потому данные вносятся в него на этапе VI.

- рекомендаций по описанию элементов модели для определения показателей функционирования (п.4.2.2, рис.4.7, 4.8), и отразить в файле (в примере “Pri”).

Результат этапа V. Указаны кратности инцидентных дуг на чертеже, созданы файлы:

- временных задержек переходов (“Tim”);
- начальной разметки (“Vr”);
- приоритетов переходов (“Pri”);
- управляющих функций (при необходимости).

Этап VI. *Определение матриц инцидентности. Проверка корректности модели.*

Исходные данные:

- чертеж графа полной сетевой модели.

Описание:

1. Произвести сканирование чертежа СП полной сетевой модели при помощи специализированной программы (приложение 2).

2. При отсутствии ошибок и корректности — к шагу 3., иначе — определить их местонахождение и устранить. Для этого использовать вспомогательные программы поиска элементов и др., послойный визуальный анализ СП. После устранения ошибок — повторить шаг 1.

3. Отредактировать созданный файл “Gen” по данным этапа V, шаг 1.

Результат этапа VI. Созданы файлы:

- матриц прямых инцидентностей (“pppp”);
- матриц обратных инцидентностей (“dddd”);
- вероятностей выходов ЖГСЧ (“Gen”).

Построение модели окончено.

Этап *VII. Испытание модели. Обработка результатов.*

Исходные данные:

- файлы “pppp”, “dddd”, “Gen”, “Pri”, “Vr”, “Uf” (при необходимости), полученные на предыдущих этапах,
- значение интенсивности потока поступающих вызовов %,

- значение допустимой вероятности рдоп попадания более 2 вызовов за временной шаг моделирования At.

Описание:

1. .Ввести значения % и $р_{доп}$.

2. Запустить программу имитационного моделирования (см. приложение 3).

3. Задать режим контроля интенсивности потока освобождений.

4. Оценить время работы модели до установившегося режима ЧНН.

5. Задать время испытаний модели, интервалы фиксации вектора разметки сетевой модели. Запустить программу.

6. По окончании моделирования считать данные фиксации вектора разметки. По полученным разметкам вычисляющих позиций определить требуемые показатели функционирования.

Результат этапа VII. Определены требуемые показатели функционирования моделируемого УК.

ВЫВОДЫ

В разделе 4 подведен итог разработки метода моделирования цифровых систем коммутации. После анализа проблемы, формулировки требований к методу и постановки задачи исследования в разделе 1 проведено исследование и разработаны принципы функционального и структурного описания цифровых систем коммутации в разделе 2, а в разделе 3 найдены способы и приемы верификации предложенных моделей. В данном разделе, для реализации процесса их отладки на основе экспериментальных исследований проведена классификация потенциальных ошибок, разработаны алгоритмы и написаны программы, для их устранения, а также для преобразования графической информации исходных данных в числовую (приложения А и Б).

Для создания универсальной имитационной моделирующей программы и реализации принципов задания динамических параметров модели предложена общая концепция модели, а также принципы задания состава и вычисления требуемых показателей функционирования УК.

Обобщены особенности сетей, создаваемых на этапе функционального и структурного описания УК, рассмотрены возможные альтернативы динамики перемещения заявок в сети. Предложено реализовать отсчет времени по принципу At, масштаб меток принять по принципу 1метка=1 заявка, статистику ВО задавать в виде ступенчатой функции плотности вероятности (гистограммы), определяющей работу ЖГСМ, для задания начальной разметки ресурсных позиций процессоров и микроЭВМ использовать приведенные оценки производительности.

В соответствии сданной концепцией разработан универсальный имитационный моделирующий алгоритм, реализующий работу СП по данным правилам. По алгоритму написана и отлажена программа (приложение В)

Рассмотрены основные качественные показатели, характеризующие работу УК: в системе с явными потерями: вероятности потерь по поступающим вызовам, по времени, по нагрузке; в системе с ожиданием: вероятность ожидания по вызовам, по времени, сверх допустимого времени по вызовам, среднее время ожидания по поступившим вызовам, по задержанным вызовам, средняя длина очереди заявок на некоторой стадии обслуживания, вероятность превышения допустимой длины очереди на стадии обслуживания, вероятность превышения допустимого времени ожидания сигналов “ответ станции” и “контроль посылки вызова” (времени задержки при коммутации); при дисциплине с повторными вызовами: среднее число попыток обслуживания; пропускная способность. Предложено задание самих показателей производить графически — условно-событийным сетевым описанием, в соответствии с содержательным смыслом величин, которые определяются. Разработаны приемы такого описания, рассмотрены характерные примеры.

Для проверки обоснованности и достоверности предложенного метода проведено построение и испытание модели цифрового узла коммутации DX-200 емкостью 2048 номеров в реальной конфигурации (приложение Д.2) при постановке задачи оценки избыточности ресурсов подсистемы управления при следующих исходных данных:

- заданная пропускная способность — 0.16 Эрл/АЛ,
- число АЛ — N=32x64=2048,
- число каналов тонального генератора “ОС” — 32,
- число каналов тонального генератора “КПВ” — 32,

- число каналов тонального генератора “СЗ” — 32,
- число приемников набора номера — 64,
- приведенная оценка микроЭВМ центрального ЗУ — 16 вызовов,
- приведенная оценка микроЭВМ маркера — 20 вызовов,
- приведенная оценка микроЭВМ регистра — 64 вызова,
- приведенная оценка микроЭВМ УУ АМ — 32х4=32 вызова,
- приведенная оценка микроЭВМ УУ БАИ — 32х1 = 128 вызовов.

Поскольку ресурсы числа каналов ИКМ (32х30=960) и УУ АМ включены неполнодоступно, произведен их пересчет в эквивалентный групповой: соответственно 548 каналов и 18 вызовов. Для данных условий составлена модель и проведено моделирование. В результате определено, что избыточность ресурсов составила 41 — 66 %, максимальная избыточность микроЭВМ УУ БАИ: необходимый ресурс составляет 44 вызова против 128 в конфигурации (66 %), а минимальный у тонального генератора “ОС” — 19 против 32 (41%). Данные выводы подтверждаются многими практическими ситуациями, когда, например^ исключение двух блоков регистра (избыточность 56%) практически не сказывается на статистике потерь.

Для иллюстрации задач метода в приложении Д.2 рассмотрены вопросы его использования в целях оптимизации.

ВЫВОДЫ

В диссертационной работе была поставлена общая задача разработки принципов и общего метода моделирования проектируемых и существующих цифровых узлов коммутации произвольной структуры для технического прогнозирования их основных качественных показателей при функционировании в условиях современных сетей связи. С целью обоснования разработки метода моделирования цифровых УК, в разделе 1 рассмотрены основные задачи стоящие перед участниками рынка коммутационного оборудования, рассмотрены известные методы моделирования УК. Анализ известных методов моделирования показал, что традиционные методы, разработанные ранее, не позволяют в полной мере решать эти задачи для современных УК, потому что: создание приемлемых аналитических моделей для цифровых УК со сложной аппаратно-программной структурой практически не возможно; традиционные методы расчета пропускной способности систем коммутации ориентированы на фиксированную структуру УК, что затрудняет решение задачи ее оптимизации; методы моделирования микропроцессорных систем общего назначения не учитывают особенностей коммутации, предполагают создание моделирующих программ для всякой новой структуры УК, что затрудняет инженерное использование их специалистами коммутационной техники.

Это позволило сформулировать требования к УК. Анализ математических схем для моделей показал, что поставленным требованиям в наилучшей степени отвечают сети Петри. Такой выбор позволил конкретизировать концепцию метода: специализация по системам коммутации; математический аппарат — СП; исходные данные — функциональная спецификация, решения этапа системного проектирования, статистика ВО; состава показателей функционирования УК задается исследователем; метод должен предусматривать проверку конструктивности реализуемого им алгоритма; модель есть набор исходных данных для использования их *универсальным* имитационным моделирующим алгоритмом; содержание модели — граф СП и числовая информация о динамике ее работы: начальная разметка, временные параметры и др.; необходимо автоматическое преобразование графической части исходных данных в цифровую информацию для моделирующего алгоритма; необходима разработка методов анализа СП с целью автоматизации проверки правильности и отладки моделей.

В соответствии с поставленной задачей, в разделе 2 разработаны принципы функционального описания цифровых систем коммутации. Принятое в разделе 1 решение о составе исходных данных и использовании СП в качестве математической схемы разрабатываемого метода моделирования определило способ формального описания УК и его внешнего окружения на базе автоматной модели SDL. При это описание ведется в виде создания отдельных подсетей: подсеть алгоритма работы УК — соответствует программе управляющего автомата модели SDL, подсеть внешнего окружения УК — соответствует операционному автомату с источником заявок, подсеть ресурсов управления УК — соответствует памяти управляющего автомата . Каждая подсеть создается отдельно, и определяется соответственно SDL-диаграммой, статистикой ВО, выбранной реализацией УК. Подсети создаются отдельно, после чего они объединяются в общую сетевую модель за счет отношений инцидентности с переходами подсети алгоритма. Учет дополнительных факторов влияния предложено реализовать введением подсети влияний, в которой размещаются ЖГСМ и соответствующие переходы. Результатом формального описания по разработанным принципам является графическая часть общей модели УК.

j В соответствии с поставленной в работе задачей, в разделе 3 проведен поиск методов алгоритмического и количественного анализа сетевых моделей, методика построения которых разработана в разделе 2. Для определения состава анализируемых свойств дана смысловая интерпретация основных алгоритмических проблем СП в соответствии с содержательным смыслом сетевых моделей УК. Определено, что свойства *ограниченности, непротиворечивости (детерминированности), живости (потенциальной живости), терминальности* являются необходимыми условиями правильности построения сетевой модели и выбора ее динамических параметров, признаком конструктивности реализуемых УК алгоритмов.

Исследование разрешимости алгоритмических проблем, соответствующих данным свойствам показало, что проблема терминальности строго разрешима, а остальные разрешимы лишь частично, на основе рассмотрения разрешимости более широкой одноименной проблемы для обычной СП совпадающей топологии. Тем не менее, данный результат позволяет оценить свойства предложенного расширения и дает возможность использования результатов классической теории СП.

Для нахождения критериев верификации сетевых моделей использован топологические ограничения сетевых моделей. В работе предложен , прием дальнейшего разделения сетевой модели на фрагменты по функциональному признаку — функциональной декомпозиции. Фрагменты выбираются так, чтобы:

- подчиняться одному из известных топологически ограниченных подклассов,
- достаточно полно . - представлять включающую его подсеть,
- выделенные фрагменты покрывали всю сетевую модель.

Предложено рассматривать автоматную подсеть алгоритма; ординарную, простую и чистую подсеть ВО, строго цикличные фрагменты подсети ресурсов. Несложные преобразования позволяют получить и автоматную подсеть ВО. Анализ данных фрагментов позволил сформулировать и доказать ряд практических критериев верификации моделей. Рассмотрены также известные методы анализа СП — уравнений состояния сети и построения дерева достижимости. Определено, что они имеют ряд недостатков, исключающих их применение для анализа сложных сетевых моделей, однако они могут использоваться для анализа относительно простых моделей частных алгоритмов на этапе вычисления временных задержек переходов алгоритма.

Рассмотрение известных методов количественного анализа СП позволило установить, что альтернативы имитационному моделированию нет. Все принятые расширения СП легко реализуются языками программирования, метод нагляден имеет удовлетворительное быстродействие. Известный метод разложения СП на инвариантные и состоятельные подсети требует почти таких же ограничений, как и СМО, применим только к относительно несложным временным СП с задержками в позициях, и может использоваться только для определения временных задержек частных алгоритмов.

В разделе 4 подведен итог разработки метода моделирования цифровых систем коммутации. Для реализации процесса отладки моделей на основе экспериментальных исследований проведена классификация потенциальных ошибок, разработаны алгоритмы и написаны программы, для их устранения, а также для преобразования графической информации исходных данных в числовую.

Для создания универсальной имитационной моделирующей программы и реализации принципов задания динамических параметров модели предложена общая концепция модели, а также принципы задания состава и вычисления требуемых показателей

функционирования УК. В соответствии с данной концепцией разработан универсальный имитационный моделирующий алгоритм, реализующий работу СП по данным правилам. По алгоритму написана и отлажена программа.

Рассмотрены основные качественные показатели, характеризующие работу УК. Предложено задание самих показателей производить графически — условно-событийным сетевым описанием, в соответствии с содержательным смыслом величин, которые определяются. Разработаны приемы такого описания, рассмотрены характерные примеры.

Для проверки обоснованности и достоверности предложенного метода проведено построение и испытание модели цифрового узла коммутации DX-200 емкостью 2048 номеров в реальной конфигурации (приложение Д.2) при постановке задачи оценки избыточности ресурсов подсистемы. В результате определено, что избыточность ресурсов составила 41 — 66 % . Данные выводы подтверждаются практическими ситуациями, когда, например исключение двух блоков регистра (избыточность 56%) практически не сказывается на статистике потерь. Для иллюстрации задач метода в приложении Д.2 рассмотрена его опросы использования в целях оптимизации.

Научное значение работы заключается:

- в развитии принципов функционального и структурного описания АО и ПО цифрового УК и процесса его работы, а именно: принципа трансляции SDL-диаграмм в графы СП, принципа воссоздания взаимодействия объектов внешнего окружения с узлом коммутации в образах СП, правил отображения ресурсных отношений элементов подсистемы управления, принципов описания различных дополнительных факторов, оказывающих влияние на работу УК;

- в исследовании применимости известных подклассов и расширений СП к функциональному и структурному описанию УК, которое привело к отысканию нового расширения сетей Петри, удобного как для описания, так и для сетевого моделирования УК;

- в исследовании применимости существующих методов качественного анализа сетей Петри к найденному расширению СП и разностороннем его исследовании, что позволило создать инструментарий для предварительной проверки корректности и отладки сетевых моделей до их испытания;

- в развитии принципов реализации сетевых моделей на основе СП на ЭВМ путем их функциональной декомпозиции, расслоения, наглядного представления с помощью современных графических сред, автоматизации процесса получения исходных данных для моделирующей программы посредством программной обработки графического изображения сетевой модели.

По результатам работы можно сделать следующие основные выводы.

1. В отличие от традиционных методов моделирования, разработанный в работе метод позволяет разделить процессы построения модели и написания программы. При этом все особенности конкретных оригиналов отражаются в графическом изображении (чертеже) модели, а сама моделирующая программа может быть универсальной. Это дает возможность инженерного использования разработанного метода моделирования специалистами в области коммутационной техники, от которых не требуется навыков программирования, а всего лишь умение построить по данному методу чертеж модели, собрать и подготовить в простейших текстовых файлах исходные данные.

2. Разработанный метод моделирования позволяет строить и анализировать модели высокой степени подробности описания оригинала вследствие иерархичности сетевых моделей на основе СП. В практической реализации любой переход сети может быть интерпретирован как подсеть, выполненная на чертеже в отдельном слое или в другом масштабе.

3. Объем задачи моделирования определяется в основном числом переходов сети(в меньшей степени числом позиций), интенсивностью поступающей нагрузки, требуемой точностью моделирования и ограничивается быстродействием используемого компьютера. Указанные параметры модели зависят от емкости УК, сложности его внутренней организации, степени подробности описания. Пути повышения объема решаемой задачи — минимизация числа переходов, особенно в подсети ВО; применение быстродействующих компьютеров; использование для написания универсальной моделирующей программы современных языков программирования высокого уровня с компиляторами оптимизированными по быстродействию; использование языков программирования низкого уровня, особенно для реализации основного цикла про-

граммы; параллельное одновременное испытание модели на нескольких ЭВМ с различными начальными установками ГСЧ.

4. Благодаря полученным в работе результатам, возможно организовать автоматическую проверку корректности модели в процессе или после программной обработки чертежа. Реализация принципа расслоения сети на сопрягаемые подсети позволяет разрешить противоречие между сложностью и наглядностью чертежа. Указанные факторы значительно облегчают процесс отладки сложных моделей, что уменьшает время, затрачиваемое на получение технического прогноза УК. Рекомендовано для создания и отладки чертежа сетевой модели использовать графические системы со встроенным языком программирования, например использованный в данной работе пакет программ AutoCAD [89].

5. Разработанный метод обеспечивает возможность модификации сетевой модели путем простого редактирования ее чертежа и перенастройки на различные исходные данные редактированием текстовых файлов. При этом значительно упрощается исследование моделируемой системы в различных специфических условиях функционирования. Так, например, учет повторных вызовов реализуется не пересчетом параметров потока вызовов в программе, а простым добавлением одной позиции ЖГСМ с соответствующими инцидентностями и заданием вероятности повторного вызова.

6. Рекомендовано использование разработанного метода имитационного моделирования цифровых узлов коммутации в эксплуатирующих их государственных и коммерческих организациях с целью технической экспертизы проектных и коммерческих решений при выборе нового оборудования, уточнении его конфигурации под конкретные условия работы, в проектных организациях для уточнения состава оборудования в технико-экономических обоснованиях и рабочих проектах по модернизации существующих и созданию новых сетей связи, в проектно-конструкторских организациях — как инструмент оценки качества проектных решений при разработке новых и модернизации существующих цифровых УК, а также как элемент САПР АО и ПО цифровых УК.

СПИСОК ИСПОЛЬЗОВАННЫХ источников

1 .Аваков Р.А., Данилов В.И., Сафронов В.Д. Зарубежные цифровые системы коммутации. - Л.: ЛЭИС, 1988. - 72 с.

2 .Артемьев М.Ю, Самоделов В.П. Программное обеспечение управляющих систем электросвязи: Учебник.- М.: Радио и связь, 1984. - 270 с.

3 .Баркун М.А. Цифровые автоматические телефонные станции: Учеб, пособ. для вузов по спец. "Автомат, электросвязь". - Минск: Вышейшая школа, 1990. - 191 с.

4 .Башарин Г.П., Бочаров П.П., Коган Я.А. Анализ очередей в вычислительных сетях. - М.: Наука 1989. - 336с.

6 .Безир Х., Хойер П., Кеттлер Г. Цифровая коммутация: Пер. с нем. - М.: Радио и связь, 1984,- 264с.

7 .Беклемишев Д.В. Курс аналитической геометрии и линейной алгебры: Учебное пособие для вузов. - М.: Наука 1976. - 320с.

8 .Берлин А.Н. Алгоритмическое обеспечение АТС. - М.: Радио и связь, 1986. - 125 с.

9 .Боккер П. ISDN. Цифровая сеть с интеграцией служб: Понятия, методы, системы: Пер. с нем. - М.: Радио и связь, 1991. - 300 с.

10 Бусленко Н.П. Моделирование сложных систем. - М: Наука, 1978. - 400 с.

Ю.Бутыльский Ю.Т. Микропроцессоры в технике связи: Учебное пособие. - Л.: ЛЭИС, 1984. -71 с.

11 .Васильев В.В., Кузьмук В.В. Сети Петри, параллельные алгоритмы и модели многопроцессорных систем. - К.: Техніка, 1990. - 212 с.

12 .Васильев Е.К., Симкин Л.М. Квазиэлектронные и электронные телефонные станции: Учебное пособие для рабочих связи. - М.: Радио и связь, 1991.- 240 с.

13 .Вычислительные комплексы и моделирование сложных систем / Под. ред. чл.-корр. АН СССР Л.Н. Королева, П.С. Краснощекова.- М.: Изд-во МГУ, 1989. - 215 с.

14 .Гилой В.К. Интерактивная машинная графика: Структура данных, алгоритмы, языки / Пер с англ. - М.: Мир, 1981. - 380 с.

15 .ГОСТ 21.835-84. Устройства коммутационной техники связи управляющие. Термины и определения. - М.: Изд-во стандартов, 1984. - 48 с.

16 .Губин Н.М., Матлин Г.М., Качество связи: Теория и практика. - М.: Радио и связь, 1986. - 272 с.

1.1 Ершов В.А. Коммутация на интегральной цифровой сети связи. - М.: Связь, 1978. - 256 с.

1.2 Ершова Э.Б., Ершов В.А. Цифровые системы распределения информации. - М.: Радио и связь, 1983. - 213 с.

19 .Жожикашвили В.А., Вишневский В.М. Сети массового обслуживания. Теория и применение к сетям ЭВМ. - М.: Радио и связь, 1982. - 182 с.

20 .Журавель В.О. Про графи досяжних розміток деяких розширень мереж Петрі // Збірник наукових праць Київського інституту залізничного транспорту. - К.: КІЗТ. - 1999. - №2.-С. 173-176.

21 .Журавель В.О. Про оцінку розв'язуваності алгоритмічних проблем для деяких розширень мереж Петрі // Мережі і системи телекомунікацій на залізничному транспорті *І* Міжвузівськ. зб. наук, пр.- Вип. 35.- Харків: ХарДАЗТ - 1999. - С. 45-49.

22 .Зайченко Ю.П., Гонта Ю.В. Структурная оптимизация сетей ЭВМ. - К.: Техніка, 1986,- 169 с.

23 .Иносэ Х. Интегральные цифровые сети связи: Введение в теорию и практику: Пер. с англ. / Под ред. В.И. Неймана. - М.: Радио и связь, 1982. - 320с.

24 .Ионин Г.Л., Седол Я.Я. Статистическое моделирование систем телетрафика. - М.: Радио и связь, 1982. - 182 с.

25 .Калашников В.В. Организация моделирования сложных систем. - М.: Знание, 1982. - 151 с.

26 .Книгавко Н.В., Журавель В.А. Моделирование внешнего окружения коммутационных систем с использованием сетей Петри *И* Информационно-управляющие системы на железнодорожном транспорте. - 1998. - №2. - С.45-50.

27 .Книгавко Н.В., Журавель В.А. Функциональное описание процесса управления коммутацией на основе сетей Петри // Информационно-управляющие системы на железнодорожном транспорте. - 1997. - №2.- С.25-31.

28 .Книгавко Н.В., Журавель В.О. Мережевий опис структур керуючих пристроїв централізованих вузлів комутації // Мережі і системи телекомунікацій на залізничному транспорті / Міжвузівськ. зб. наук, пр.- Вип. 35. - Харків: ХарДАЗТ. - 1999. - С. 27-36.

29 .Кнут Доналд Е. Искусство программирования на ЭВМ: В 7 т.: Пер. с англ.- М.: Мир, 1977. - Т. 2: Получисленные алгоритмы. - 728с.

30 .Кожанов Ю.Ф. Расчет и проектирование электронных АТС: Справочник. - М.: Радио и связь, 1991. - 144 с.

31 .Коммутация и управление потоками в сетях связи: Сб. нучн. тр. ин-тов связи / Редкол. М.А. Сиверс и др. - Л.: ЛЭИС, 1987. - 176 с.

32 .Концепція побудови цифрової мережі зв'язку залізничного транспорту (в порядку обговорення). К.: Укрзалізниця, 1998. - 28 с.

33 .Корнышев Ю.Н., Фань Г.Л. Теория распределения информации: Учеб, пособие для ву-зов. - М.: Радио и связь, 1985. - 184 с.

34 .Котов В.Е. Сети Петри. - М: Наука, 1984. - 160 с.

36 .Котов В.Е., Сабельфельд В.К. Теория схем программ. - М.: Наука, 1991. - 248 с.

Зб.Кристофидес Н. Теория графов: Алгоритмический подход: Пер с англ. - М.: Мир, 1978.-432 с.

37 .Крон Г. Исследование сложных систем по частям (диакоптика): Пер. с англ.- М.: Наука, 1972. - 544 с.

38 .Кудрявцев В.Г., Воронин Л.Е., Поляк М.И. Новая концепция построения интегральной цифровой сети. Техника, экономика, стандартизация//Электросвязь.- 1991.- №1,- С.2-11.

39 .Лазарев В.Г., Маркин Н.П., Лазарев Ю.В. Проектирование дискретных устройств автоматики: Учеб, пособие для вузов связи. - М.: Радио и связь, 1985. - 168с.

40 .Лазарев В.Г., Пийль Е.И., Турута Е.Н. Построение программируемых управляющих устройств. - М.: Энергоатомиздат, 1984. - 192с.

41 .Лазарев В.Г., Пийль Е.И., Турута Е.Н. Программное управление на узлах коммутации. - М.: Связь, 1978. - 264с.

42 .Лутов М.Ф., Жарков М.А., Юнаков П.А. Квазиэлектронные и электронные АТС. М.: Радио и связь, 1988. - 264 с.

43 .Майника Э. Алгоритмы оптимизации на сетях и графах: Пер с англ. - М.: Мир, 1981. - 323 с.

44 .Мартин Дж. Вычислительные сети и распределенная обработка данных: Программное обеспечение, методы и архитектура. Вып. 2: Пер. с англ. - М.: Финансы и статистика, 1986 г. - 269 с.

45 .Микропроцессорные средства обработки и отображения информации в системах управления и связи: Сб. ст. / Под ред. В.Н.Соловейчика- М.: Радио и связь, 1983. - 158с.

46 .Минский М.А. Вычисления и автоматы.: Пер. с англ. - М.: Мир 1984. - 364 с.

47 .МККТТ, Оранжевая книга. - М.: Связь 1979.- Т.VI-4: Языки программирования для телефонных станций с программным управлением. - 64с.

48 .Модели и методы исследований в системах информатики: Сб. ст. / Ин-т проблем пер. информации / Отв. ред. А.Д.Харкевич, В.А. Гармаш.- М.: Наука, 1988. - 156 с.

49 .Моделирование вычислительных систем и процессов: Межвузовск. сб. научн. тр. / Пермск. гос. ун-т им. А.М. Горького. - Пермь: БИ, 1990. 129 с.

50 .Моделирование и управление в гибких автоматизированных производствах и системах автоматического управления: Межвузовск. сб. научн. тр. / Моск, ин-т радиотехники, электроники и автоматики. - М.: МИРЭА, 1990. - 142 с.

51 .Моделирование процессов обработки информации и управления: Межвед. сб. / Моск, физ.-техн. ин-т. - М.: МФТИ, 1990. - 171 с.

52 .Моделирование систем и процессов связи: Сб. научн. тр. учебн ин-тов связи / Ленингр. электро-техн, ин-т связи им. М.А. Бонч-Бруевича. - Л: ЛЭИС, 1988 - 170 с.

53 .Молчанов А.А. Моделирование и проектирование сложных систем. - К.: Вища школа, 1988. - 360 с.

54 .Назаров А.Н., Симонов М.В. АТМ: Технология высокоскоростных сетей / Серия "Инженерная энциклопедия ТЭК - технологии электронных коммуникаций". - М.: ИТЦ "ЭКО-ТРЕНДЗ"- 1998. - 573 с.

55 .Нейман В.И. Теоретические основы Единой автоматизированной сети связи. - М.: Наука, 1984. - 244 с.

56.Оборудование DX-200.- Проспект фирмы Nokia, 1984. - 29 с.

57 .Петренко А.И. Семенков О.И. Основы построения систем автоматизированного проектирования: Учебник для инж. спец, вузов. - К.: Техніка 1984. - 294 с.

58 .Пийль Е.И. Описание процессов в SDL и на языке сетей Петри *И* Управление в распределенных интегральных сетях. - М.: Наука 1991. - С.30-37.

59 .Питерсон Дж. Теория сетей Петри и моделирование систем: Пер. с англ. - М.: Мир, 1984. - 265 с.

60 .Полляк Ю.Г., Филимонов Ю.А. Статистическое машинное моделирование средств связи. - М.: Радио и связь, 1988. - 176 с.

61 .Пранявичус Г.И. Модели и методы исследования вычислительных систем. - Вильнюс: Мокслас, 1982. - 228 с.

62 .Применение микропроцессорных средств в системах передачи информации: Учеб, пособ. для вузов по спец. АСУ / Б.Я.Советов, О.И.Кутузов, Ю.А.Головин, Ю.В.Аветов. - М.: Высшая школа, 1987. - 254 с.

63 .Программирование микропроцессорных систем / В.Ф. Шаньгин, А.Е. Костин, В.М. Илю-шечкин, П.А. Тимофеев / Под ред. В.Ф. Шаньгина.- М.: Высшая школа, 1990-303с.

64 .Райс Дж. Матричные вычисления и математическое обеспечение: Пер. с англ. - М.: Мир, 1984,-264 с.

65 .Системы параллельной обработки / Ж.-Л. Баер, Р. Барлоу, М. Вудворд и др. / Под ред. Д. Ивенс : Пер. с англ.- М: Мир, 1984. - 413 с.

66 .Слепцов А.И., Юрасов А.А. Автоматизация проектирования управляющих систем гиб-ких автоматизированных производств. - К.: Техніка. 1986. - 138 с.

67 .Советов Б.Я., Яковлев С.А. Моделирование систем: Учебник для вузов. - М.: Высшая школа, 1998. - 319с.

68 .Советов Б.Я., Яковлев С.А. Построение сетей интегрального обслуживания. - Л.: Машиностроение, 1990. - 331 с.

69 .Средства коммутационной техники для цифровых сетей с интеграцией служб / Аналитический обзор по материалам зарубежной печати / сост. А.Я. Усков. - М.: ЦООНТИ "ЭКОС", 1987. - 68с.

70 .Стародубцев Н.А. Синтез схем управления параллельных вычислительных систем / Отв. ред. Н.К. Златорунский.- Л.: Наука, 1984. - 191 с.

71 .Теория систем и методы системного анализа в управлении и связи / В.Н. Волкова, В.А. Воронков, А.А. Денисов и др. - М.: Радио и связь, 1983. - 248 с.

72 .Теория телетрафика в системах информатики: Сб. научн. тр. - М.: Наука, 1989 - 152 с.

73 .Территориально-производственные комплексы: Прогнозирование процесса формирования с использованием сетей Петри / М.К. Бандман, О.Л. Бандман, Т.Н. Есипова и др. / Отв. ред. А.Г. Грандберг. - Новосибирск: Наука, 1990. - 297 с.

74 .Технология системного моделирования / Е.Ф. Аврамчук, А.А. Вавилов, С.В. Емельянов и др. / Под общ. ред. .В. Емельянова. - М.: Машиностроение; Берлин: Техника, 1988. - 520 с.

75 .Турбо Паскаль 7.0: Учебное издание. - К.: Издательская группа BHV, 1996. - 448 с.

76 .Управление ресурсами в интегральных сетях: Сб. науч. тр. / АН СССР Ин-т пробл. передачи инф. / Отв. ред. В.Г. Лазарев, В.Г. Черняев. - М.: Наука, 1991. - 140 с.

77 .Управляющие системы электросвязи и их программное обеспечение: Учебник для вузов / Р.А.Аваков, В.О.Игнатьев, А.Г.Панова, Н.С.Чагаев. - М.: Радио и связь, 1991.- 256 с.

78 .Фельдман Л.П., Слепцов А.И., Дедищев В.А. Оптимизация структур и процессов в вычислительных системах методами имитационного моделирования. - Донецк: Донецкий политехи, ин-т, 1978. - 79 с.

79 .Фоли Дж., вэн Дэм А., Основы интерактивной машинной графики: В 2-х кн. Кн. 1: Пер. с англ. - М.: Мир, 1984. - 368 с.

80 .Фрей Д. AutoCAD 14 на примерах: Пер. с англ. - К.: ЮНИОР, 1998. - 560 с.

81 .Фридмен М., Ивенс Л. Проектирование систем с микрокомпьютерами: Пер. с англ. - М.: Мир, 1986. - 405 с.

82 -Хиллс М.Т. Принципы коммутации в электросвязи: Пер. с англ. / Под ред. В.И. Неймана. - М.: Радио и связь, 1984. - 312с.

83 .Хоббс М. Современные системы коммутации в электросвязи: Пер. с англ. - М.: Связь, 1978. - 328 с.

84,Цифровая и вычислительная техника: Учебн. для вузов по спец. "Автомат, электросвязь" / Э.В. Евреинов, Ю.Т. Бутыльский, И.А, Мамзелев и др. / Под ред. Э.В. Евреинова. - М.: Радио и связь, 1991. - 463 с.

85 .Цифровая связь: справочник / И.П. Панфилов, В.К. Стеклов, И.Л. Бирюков и др. / Под ред. В.К. Стеклова. - К.: Техніка, 1992. - 230 с.

86 .Цифровые устройства и микропроцессоры в системах информации: Сб. ст. / Под общ. ред. П.Ф.Полякова.- Харьков: ХИИТ, 1987.- 100 с.

87 .Черняев В.Г. Вопросы композиции сетей Петри, ч.П //Управление в распределенных интегральных сетях. - М.: Наука 1991. - С.37-48.

88 .Чуркин В.П. Асинхронные цифровые системы коммутации. - М.: Радио и связь, 1984. - 191 с.

89 .Чуркин В.П. Развитие систем распределенного управления зарубежных АТС *И* Техника средств связи. Сер. ТПС. - 1985. - Вып.6. - С. 38 - 47.

90 .Шастова Г.А., Коекин А.И. Выбор и оптимизация структуры информационных систем. - М.: Энергия, 1972. - 256с.

91 .Шнепс М.А. Системы распределения информации. Методы расчета М.: Связь, 1979. - 342 с.

92 .Штагер В.В. Цифровые системы связи: Теория, расчет и оптимизация. - М.: Радио и связь, 1993. - 309с.

93 .Штагер В.В. Электронные системы коммутации. - М.: Радио и связь, 1983. - 231 с.

94 .Электронные цифровые системы коммутации: Учеб пособие для вузов / Н.Ф.Болгов, Т.П.Гуан, О.А.Соболев, А.В.Танько. - М.: Радио и связь, 1985. - 144 с.

95 .Элементы параллельного программирования / В.А.Вальковский, В.Е.Котов, А.Г.Марчук, Н.Н. Миренков / Под ред. В.Е. Котова. - М.: Радио и связь, 1983. - 240 с.

96 .Янбых Г.Ф., Эттингер Б.Я. Методы анализа и синтеза сетей ЭВМ. - Л.: Энергия, 1980.-95 с.

97 .Red Book. Vol. VI. 1. General Recommendations Telephone Switching and Signalling. Interface with the Maritime Mobile and the Land Mobile Services. Recommendations Q.l—Q.l 18bis // CCITT VIII Plenary Assembly (Malaga-Torremolinos, 8-19 October 1984). - Geneva: ITU, 1985,- 320 p.

98 .Red Book. Vol. VI.5. Digital Tranzit Exchanges in Integrated Digital Networks and Mixed Analogue-Digital Networks. Digital Lockal and Combined Exchanges. Recommendations Q.501—Q.517. Ibid.- 1985.- 132 p.

99 .Red Book. Vol. VI. 10. Functional Specification and Description Language (SDL). Recommendations Z.101—Z.104. Ibid.- 1985.- 134 p.

100 .Red Book. Vol. VI. 10. Functional Specification and Description Language (SDL). Annexes to Recommendations Z.101— Z.104. Ibid.- 1985.- 271 p.

ПРИЛОЖЕНИЕ А

Программы построения и анализа дерева достижимости сетей

Для построения и анализа дерева достижимости с ограничением ветвей ©-листами (ПД) написаны программа **“P_tree”** и **“Explorer”.** Ниже следуют пояснения к ним.

В программе “P_tree” для построения дерева достижимости (ПД) исходными данными являются текстовые файлы с данными: матриц инцидентности, вектора приоритетов (целые неотрицательные числа), вектора задержек переходов, вектора начальной разметки. Можно не задавать вектор приоритетов, задержек, или оба вектора. При этом строится дерево соответствующей СП. Концепция построения дерева соответствует изложенной в п.3.1.4. Считается, что СП синхронна, т.е. в каждом такте обязательно срабатывают возбужденные переходы, причем по правилам приоритетной СП.

Общий алгоритм программы представлен на рис. А. 1.

Вершине ПД в создаваемом файле соответствует набор данных, включающий собственно разметку позиций СП, ее тип, таблицу очередей задержанных (поглощенных) меток (см. рис.3.10 в п.3.1.4) в несколько модифицированном виде, ссылки на предшествующую и последующие вершины (родителя и потомка), флаги и другая служебная информация необходимая для быстрого поиска вершины и ее анализа.

Дадим пояснения к программе “P_tree”.

Начало.

При загрузке, у пользователя запрашиваются имена файлов с исходными данными (матрицы прямых и обратных инциденций, начальная разметка, а также, в качестве необязательных, имена файлов с вектором приоритетов и вектором временных задержек), которые считываются в память компьютера для проверки корректности этих данных (проверка совпадения размерностей считанных матриц) и дальнейшего использования. Кроме того производится необходимое распределение динамической памяти, присвоение переменным необходимых начальных значений, создание (открытие) файла результата и т.п.[1] После этого создается вершина дерева (в соответствии со считанными начальными данными) и делается пометка ее как текущей.

[1] Если необязательный параметр не вводится (т.е. нажимается просто Enter) то это влияет на состояние флагов типа сети (является ли она временной и/или приоритетной).

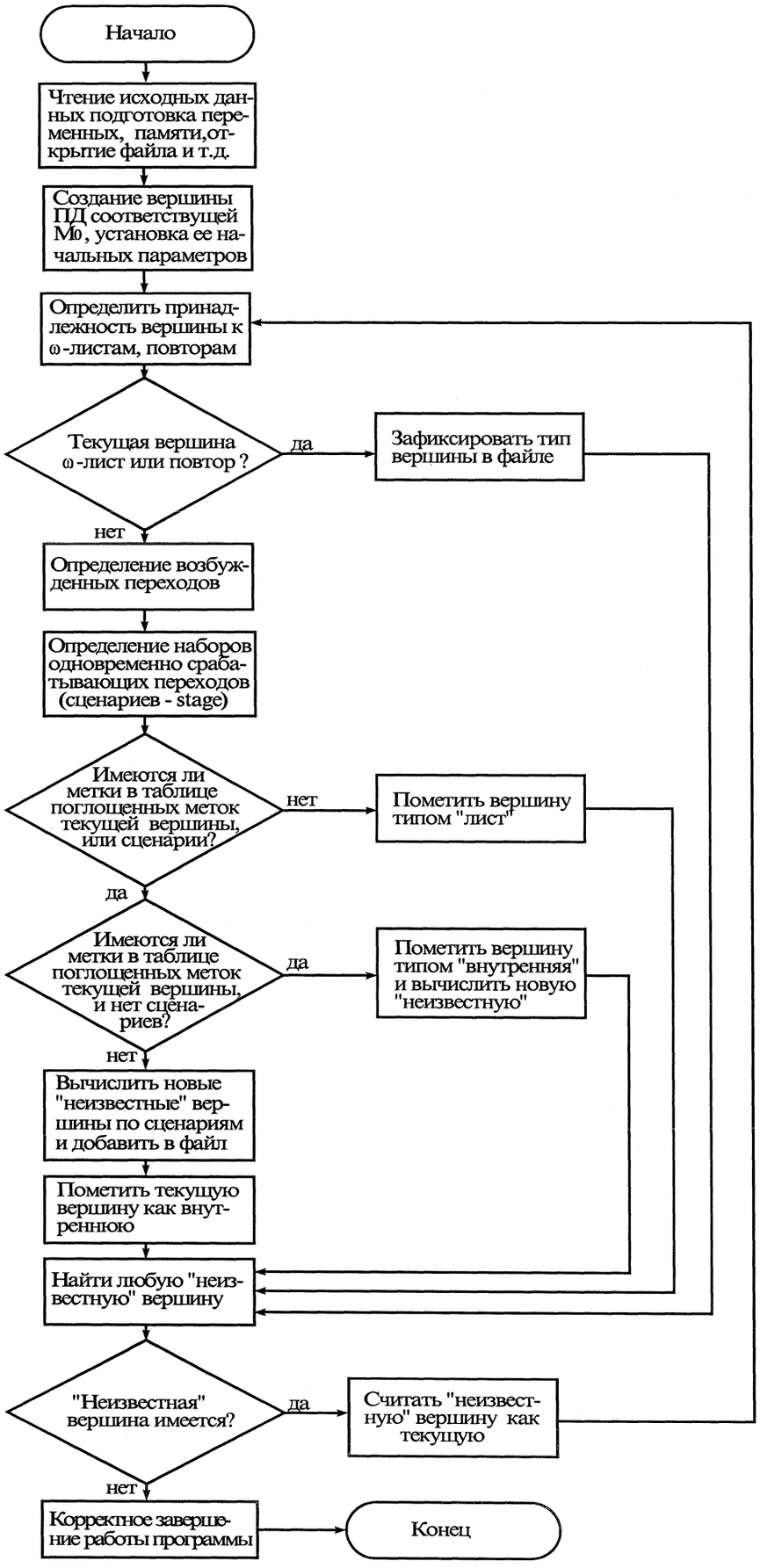


Рис.А. 1. Блок-схема алгоритма программы “P_tree

Если при вводе не задаются файлы векторов задержки и приоритетов, то флаги типа сети принимают такое значение, при котором исключается отработка соответствующих фрагментов программы с целью ускорения ее работы.

Под типом вершины понимается принадлежность вершины к следующим подмножествам:

1. Внутренние вершины ПД имеющие предшественника(-ов) —“предка” (“родителя”) и последователей — “потомков”.
2. Повторяющаяся разметка — “повтор”:

   $M' = M,$
3. ©-листы:

   $M' > M,$
4. Тупиковые разметки — отсутствие возбужденных переходов и меток в таблице очередей задержанных меток — “тупики”.
5. Вычисленные, но еще не исследованные вершины — “неизвестные”.

Проверка на лист, W-лист и повтор.

Данная функция (Search_WList) оформлена в модуле ЮТгее (файл iotree.pas), и реализована следующим образом:

1. Проверка на ю-лист[1]. Для этого производится возврат к “предку” и сравнение текущей вершины с этим “предком”. Условием остановки возвратов к предку является либо нахождение вершины, удовлетворяющей условию поиска, либо достижение корня (первой вершины) дерева. В последнем случае текущая вершина не является W-листом.

2. Проверка на повтор (то есть поиск уже существующих вершин, имеющих характеристики полностью идентичные характеристикам текущей вершины). Это сделано для сокращения объема и времени построения дерева за счет однократного просчета идентичных ветвей (после первого просчета вместо этой ветви указывается только указатель на эту ветвь). Для реализации этого производится перебор всего файла из конца к началу с поиском вершины, идентичной текущей.

Для сравнения вершин использован следующий алгоритм (функция ComparisonltemTree):

[1] В пояснениях листинга программы использована несколько отличная от общепринятой система именований: лист — разметки равны, <o -лист — первая разметка строго больше второй.

1) Сравниваются разметки (первая больше второй — ">", первая меньше второй — "=", несопоставимые разметки —"?").
2) Если сеть определена как временная, то проводится сравнение таблиц состояния переходов(по аналогичным правилам и с теми же результатами сравнения), а иначе считаем таблицы состояния переходов равными.
3) Находится общий результат сравнения по отдельным результатам сравнения разметок и таблиц состояния переходов в соответствии с табл. А. 1.

Таблица А.1

Разметки	Таблица состояния переходов	Результат
¹ > ’	¹ >¹	ı y !
’ > ¹	¹ = ’	1 > ï
’ > ’	■ ’ ? ¹	’ ? ’
’ = ’	¹ >¹	I > 1
’ = ’	’ = ’	1 — I
’ = ’	i 9 i	i 9 i
¹ ? ’	*1 > 1*	i 9 ɨ
i ? i	’ = ’	i 9 i
» ? i	’ ? ’	’ ? ¹

Поиск возбужденных переходов.

Данная функция (FindStimulationPassage) оформлена в модуле Units (файл units.pas),Н реализована следующим образом. Используя матрицу прямой инцидентности и вектор разметки определяем для перехода, достаточно ли ему меток для срабатывания (проверка идет по всем позициям), и в случае достаточности переход считается возбужденным. Такую проверку выполняем для всех переходов.

Поиск “сценариев” срабатываний.

Данная функция (FindStage) оформлена в модуле Units (файл units.pas), и реализована по следующему алгоритму:

1) установить максимальный приоритет;
2) выявить возбужденную группу переходов с установленным приоритетом;
3) таблица пуста? — если нет, то перейти к 5);

4) обнулить наборы таблицы сценариев;
5) добавить к набору 1 переход из группы;
6) определить для набора получаемую разметку;
7) определить для такой разметки, какие переходы остались возбужденными (в группе);
8) образовать наборы с возбужденными переходами;
9) найти одинаковые наборы и обнулить лишние;
10) уничтожить пустые строки в таблице сценариев.
11) если за проход таблица изменилась, то к 5);
12) если приоритет группы больше -1, то декрементировать его и перейти к 2)

Вычисление неизвестных вершин (“моделирование” по “сценарию”).

Такое моделирование проводится при добавлении вершины (в соответствии с одним из сценариев срабатывания). Данная функция (SwallowingAndDeliveranceMalk — поглощение и освобождение меток) оформлена в модуле TimeOut (файл timeout.pas), и реализована следующим образом:

1) Поглощение меток сработавшими переходами.
2) Выдача меток с истекшей задержкой всеми переходами.
3) Обновление таблицы состояний переходов.

Конец.

При завершении работы программы необходимо освободить ранее распределенную динамическую память и закрыть все открытые файлы.

Вторая программа — **“Explorer”** преназначена для исследования построенного ПД путем процедур просмотра, поиска интересующей разметки по различным критериям, ее просмотра, анализа последовательностей сработавших переходов, сопоставления разметок, нахождения предшественников (“родителей”) и последователей (“потомков”), просмотра различной дополнительной информации по любой вершине-разметке с использованием стандартных окон Windows.

Приведем некоторые пояснения к программе. “Explorer” состоит из:

- основного файла (EXPLORER.DPR);
- модуля, реализующего чтение дерева (iotree.pas);
- модуля с описанием типа элементов дерева (typetree.pas);

- модуля главной формы приложения (unit 1 .pas);
- модулей реализующих другие используемые формы (unit2.pas, unit3.pas, unit5.pas, unit6.pas, unit7.pas, unit8.pas, unit9.pas);
- модуля реализации чтения дерева и заполнения стандартного объекта OutLine полученной информацией, а также некоторых вспомогательных процедур (unit4.pas).

Организация интерфейса с пользователем основана на использовании стандартных средств, предоставляемых Windows и Delphi. Алгоритмы поиска реализованы в модулях с описанием соответствующих форм, и по существу представляют себой перебор вершин дерева с поиском по маске, заданной в форме.

Таким образом, рассмотрим только модуль unit4, так как другие модули, как и основная программа, служат лишь для представления считанного дерева стандартными средствами и объектами Delphi и интереса не представляют. В unit4 рассмотрим лишь алгоритм чтения дерева из файла.

Алгоритм чтения дерева.

1) открывается файл с деревом;
2) очищается стандартный элемент Delphi, служащий для просмотра древовидных структур — OutLine;
3) читается первая запись в файле, проверяется ее структура (в случае несоответствия — сообщение об ошибке формата файла) и определяются некоторые основные параметры дерева (число переходов СП);
4) в цикле (до конца файла) выполняем следующие действия:
   a) читаем следующую запись;
   b) для подсчета количества вершин различного типа, определяем ее тип и инкрементируем счетчик соответствующий этому типу;
   c) перебором считанной части дерева находим родителя данной вершины и добавляем ее к потомкам родителя;
   d) даем команду объекту OutLine[1] вывести прочитанное дерево на экран.

Далее следует листинги программ **“P_tree”** и **“Explorer”.**

[1] Элемент OutLine не допускает уровня вложенности более 255 и поэтому отслеживается достижение этой границы и в таком случае информация о таких вершинах в OutLine не добавляется но сохраняется в специальной структуре (для дальнейшей корректной работы, например проведения поиска по дереву).

Программа “P tree’".

```
{ Эта программа предназначена для построения дерева достижимости сети }
{ Петри по заданным матрицам, описывающим эту сеть. }
{ For Delphi 4.0                                    }
{SAPPTYPE CONSOLE}
program P_Tree;
uses KijCRTW,ReadData,Typetree,TimeOut,Units,lOTree;

{$R *.RES}
var Takt:longint; {так тработы}
   Razm:ArrayVectorOfWord; {текущая разметка}
   i,j :byte; {временные переменные}
   TempIDilongint; {текущий ID}
   TempID2:longint; {временная переменная для хранения ID}
   PTempItemTree:^TItemTree; {обрабатываемая вершина дерева}
   PTempItemTreel:^TItemTree; {вершина дерева}
   TempTypeItem:TTypeItem; {тип вершины}
   Point_ID_NE:longint;
   TempFlagzboolean;
   Temp:longint;
begin
{..... .....инициализация.....— .....}
ClrScr;
Writeln(’Строится дерево для заданной сети Петри.');
New(PTempItemTree);
New(PTempItemTree 1);
InitialiseFile(FileName);
{-------- ------- добавление в файл дополнительной информации— .....}
PTempItemTree^.NU:= 1; {версия}
PTempItemTree^.M:=M; {начальная разметка}
PTempItemTree^.P[0]:=0; {подверсия}
PTempItemTree^.OldPoint_ID:=1998; {идентификационный код}
PTempItemTree^.FlagNew:=True;
PTempItemTree^.TypeItem:=Unknown;
PTempItemTree^.W_Point_id:=P; {количество вершин-позиций}
PTempItemTree^.Point_ID_FirstDescendant:=-1;
PTempItemTree^.Point_ID_LastDescendant:=-1;
PTempItemTree^.NumberDescendant:=D; {количество вершин-переходов}
MakeUpItemTree(PTempItemTree^); {добавление элемента списка}
\Угке1п('Дополнительная информация добавлена к файлу.');
{-..... создание вершины дерева и установка начальных параметров ..... .....}
Takt:=0;
Razm:=M;
FillChar(StimulationPassage^,SizeOf(StimulationPassage^),0);
Stage^.Size:=O;
Indicate(O); {индикатор выполнения}
PTempItemlYee ^ .NU:=T&kt;
PTempItemTree^.M:=Razm;
PTempItemTree^.P[0]:=0;
PTempItemTree^.TimeTable:=TSM^;
PTempItemTree^.OldPoint_ID:=-1;
PTempItemTree^.TypeItem:=UnKnown;
PTempItemTree^.FlagNew:=False;
PTempItemTree^. W_Point_id:=-1;
PTempItemTree^.Point_ID_NE:=-1;
PTempItemTree^.Point_ID_FirstDescendant:=-1;
PTempItemTree^.Point_ID_LastDescendant:=-1;
PTempItemTree ^ .NumberDescendant:=-1;
MakeUpItemTree(PTempItemTree^);
TempID:=IDFlowingItemTree-1;
Temp:=l;
Takt:=l;
{-..... .....ОСНОВНОЙ ЦИКЛ... ..... ..... ..... }
repeat
{проверка на лист, W-лист и повтор}
TempID2:=Search_WList(PTempItemTree^,TempID,OI,TempTypeItem,P,D,FlagTimesNet);
if TempID2 >= 0 then
begin
PTempItemTree^.TypeItem:=TempTypeItem;
  PTempItemTree^.W_Point_ID:=TempID2;
```

```
PTempItemTree^A.Point_ID_FirstDescendant:=-1;
PTempItemTree^A.Point_ID_LastDescendant:=-1;
PTempItemTree^A.NumberDescendant:=O;
PTempItemTree^A.FlagNew:=True;
WriteItemTree(PTempItemTree^A,TempID);
end else
begin
FindStimulationPassage(P,D,
PI, {матрица прямых инцинденций}
PTempItemTree^A.M, {вектор разметки}
StimulationPassage^A);
FindStage(FlagPriorityNet, {флаг приоритетности сети}
        StimulationPassage^A,{табл, возбужд.переходов}
        PTempItemTree^A.M , {вектор текущей разметки}
        VP, {вектор приоритетов переходов}
        PI, {матр. прям, инцинденций}
        P,D, {размерность MPI}
        Stage^A); {резуьтат}
TempFlag:=False;
for i:=0 to D do
   for j:=0 to CvantItemMax do
     if PTempItemTree^A.TimeTable[i,j] <> -1 then TempFlag:=True;
if (Stage^A.Size = 0)and(not TempFlag) then
   begin
   PTempItemTree^A.TypeItem:=BluntKnife;
   PTempItemTree^A.W_Point_ID:=-1;
   PTempItemTree ^A .Point_ID_FirstDescendant:=-1;
   PTempItemTree^A.Point_ID_LastDescendant:=-1;
   PTempItemTree^A.NumberDescendant:=O;
   PTempItemTree^A.FlagNew:=True;
   WriteItemTree(PTempItemTree^A,TempID);
   end else
   begin
   if (Stage^A.Size = O)and(TempFlag) then
   begin
   {добавляем новую вершину(т.к.есть поглощенные метки)}
   Inc(Temp);
    TSM^A:=PTempItemTree^A.TimeTable;
    Razm:=PTempItemTree^A.M;
    SwallowingAndDeliveranceMalk( {поглощение и освобождение меток}
          P, {кол-во позиций}
          D, {кол-во переходов}
          P1,{матр. прямых инцинденций}
          01, {матр. обратных инцинденций}
          Stage^A.V[i],{номера сработавших переходов}
          TSM^A,{таблица состояний переходов}
          Razm, {вектор разметки}
          VZ, {вектор задержек}
          FlagTimesNet); {True если сеть временная}
    PTempItemTree 1 ^A.NU:=Takt;
    PTempItemTree 1^A. M:=Razm;
    PTempItemTree 1 ^A.P[0]:=0;
    PTempItemTreel^A.TimeTable:=TSM^A;
    PTempItemTreel^A.OldPoint_ID:=TempID;
    PTempItemTree 1 ^A.FlagNew:=False;
    PTempItemTreel^A.TypeItem:=Unknown;
    PTempItemTree 1^A. W_Point_id:=-1;
    PTempItemTree 1^A .Point_ID_FirstDescendant:=-1;
    PTempItemTree 1 ^A.Point_ID_LastDescendant:=-1;
    PTempItemTree 1^A .NumberDescendant:=-1;
    if i > 1 then PTempItemTree 1 ^A.Point_ID_NE:=Point_ID_NE
          else PTempItemTree 1 ^A.Point_ID_NE:=-1;
    MakeUpItemTree(PTempItemTreel^A); {добавление элемента списка}
    PTempItemTree^A.Point_ID_FirstDescendant:=IDFlowingItemTree-1;
    PTempItemTree^A.Point_ID_LastDescendant:=IDFlowingItemTree-1;
    Indicate(Temp); {индикатор выполнения}
    PTempItemTree^A.TypeItem:=Internal;
    PTempItemTree^A.W_Point_ID:=-l;
    PTempItemTree^A.NumberDescendant:= 1;
    PTempItemTree^A.FlagNew:=True;
```

```
    WriteItemTree(PTempItemTree^A,TempID);
    end else
    begin
    {создаем по сценариям новые вершины}
    for i:=l to Stage^A.Size do
     begin
     Inc(Temp);
     TSM^A:=PTempItemTree^A.TimeTable;
     Razm:=PTempItemTree^A.M;
     SwallowingAndDeliveranceMalk( {поглощение и освобождение меток}
           P, {кол-во позиций}
           D, {кол-во переходов}
           P1,{матр. прямых инцинденций}
           01, {матр. обратных инцинденций}
           Stage^A.V[i], {номера сработавших переходов}
           TSM^A, {таблица состояний переходов}
           Razm, {вектор разметки}
           VZ, {вектор задержек}
           FlagTimesNet);{True если сеть временная}
     PTempItemTree 1 ^A.NU:=Takt;
     PTempItemTree 1 ^A.M:=Razm;
     PTempItemTreel^A.P:=Stage^A.V[i];
     PTempItemTreel^A.TimeTable:=TSM^A;
     PTempItemTreel^A.OldPoint_ID:=TempID;
     PTempItemTree 1 ^A.FlagNew:=False;
     PTempItemTree 1^A .TypeItem:=Unknown;
     PTempItemTree 1^A. W_Point_id:=-1;
     PTempItemTree 1 ^A.Point_ID_FirstDescendant:=-1;
     PTempItemTree 1^A .Point_ID_LastDescendant:=-1;
     PTempItemTree 1 ^A.NumberDescendant:=-1;
     if i > 1 then PTempItemTreel^A.Point_ID_NE:=Point_ID_NE
           else PTempItemTree 1^A .Point_ID_NE:=-1;
     MakeUpItemTree(PTempItemTreel^A); {добавление элемента списка}
     Point_ID_NE:=IDFlowingItemTree-1;
     if i = 1 then PTempItemTree^A.Point_ID_FirstDescendant:=Point_ID_NE;
     if i = Stage^A.Size then PTempItemTree^A.Point_ID_LastDescendant:=Point_ID_NE;
     Indicate(Temp); {индикатор выполнения}
     end;
    PTempItemTree^A.TypeItem:=Internal;
    PTempItemTree^74. W_Point_ID:=-1;
    PTempItemTree^A .NumberDescendant:=Stage^A .Size;
    PTempItemTree^A.FlagNew:=True;
    WriteItemTree(PTempItemTree^A,TempID);
    end;
   end;
end;
TempID:=FindNewExploreItemTree( {поиск новой исследуемой вершины (-1 если не найдена)}
                TempID {номер элемента с которого начнется поиск}
                ); {номер найденого элемента или -1}
if TempID >= 0 then ReadItemTree(PTempItemTree^A,TempID); {чтение элемента списка}
Takt:=PTempItemTree^A.NU+l;
until (TempID<0)or(FindStopKij);
{-..... -.....-деинициализация.....-..... }
DeinitialiseFile;
Dispose(PTempItemTree);
Dispose(PTempItemTreel);
А¥гйе1п('Дерево построено в файле ',FileName + '.TR');
\Угйе1п('Для корректного завершения программы нажмите Enter.');
Readin;
end.
```

UNITKijCRTW; {FOR CONSOLE APPLICATION}

```
INTERFACE
procedure ClrScr;
procedure GotoXY(X,Y:Word);
function WhereXrlongint;
function WhereY:longint;
function FindStopKij:boolean; {True если завершается работа приложения}
IMPLEMENTATION
USES Windows;
VAR ConInputHandle:THandle;
```

```
   ConOutputHandleiTHandle;
   MaxCoord:TCoord;
   NOAW:Integer;
procedure ClrScr;
var Coord:TCoord;
begin
Coord.X:=0;Coord.Y:=0;
FillConsoleOutputCharacter(ConOutputHandle,' ',MaxCoord.X*MaxCoord.Y,Coord,NOAW);
GotoXY(0,0);
end;
procedure GotoXY(X,Y:Word);
var Coord:TCoord;
begin
Coord.X:=X;Coord.Y:=Y;
SetConsoleCursorPosition(ConOutputHandle,Coord);
end;
function WhereXdongint;
var TempBuffer:TCONSOLESCREENBUFFERINFO;
begin
GetConsoleScreenBufferInfo(ConOutputHandle,TempBuffer);
WhereX:=TempBuffer.dwCursorPosition.X;
end;
function WhereYdongint;
var TempBufferiTCONSOLESCREENBUFFERINFO;
begin
GetConsoleScreenBufferInfo(ConOutputHandle,TempBuffer);
WhereY :=TempBuffer.dwCursorPosition. Y;
end;
function FindStopKij:boolean; {True если завершается работа приложения}
var IBuff:TInputRecord;
   IEvent:DWord;
   c:Char;
   T empBuff: Integer;
begin
Result:=False;
GetNumberOfConsoleInputEvents(ConInputHandle,TempBuff);
if TempBuff > 0 then
begin
ReadConsoleInput(ConInputHandle,IBuff, 1 JEvent);
case IBuff.EventType of
    KEY-EVENT:
      begin
      if      ((IBuff.KeyEvent.bKeyDown      =      True)and(IBuff.KeyEvent.wVirtualKeyCode      =      VK_ESCAPE))
        then                                                                                                  Result:=True
        else Result:=False;
      if (IBuff.KeyEvent.bKeyDown = True) then FlushConsolelnputBuffer(ConlnputHandle);
      end;
end; {case}
if Result then
begin
Writein;
repeat
Writeln('{ yГаиЁвм а У®вг Iа®Ja ~~1n(Y/N):');
ReadLn(c);
until (c='Y')or(c='y')or(c='N')or(c='n');
if (c='N')or(c='n') then Result:=False;
end; {if}
end; {if}
end;
BEGIN
ConInputHandle:=GetStdHandle(STD_INPUT_HANDLE);
ConOutputHandle:=GetStdHandle(STD_OUTPUT_HANDLE);
MaxCoord:=GetLargestConsoleWindowSize(ConOutputHandle);
END.
```

UNIT READDATA;

```
INTERFACE
USES TypeTree;
```

```
VAR PE Array MatrOfWord; {матрица прямых инцинденций}
    OEArrayMatrOfWord; {матрица обратных инцинденций}
    M:Array VectorOfWord; {вектор начальной разметки}
    VP:ArrayVectorOfWord; {вектор приоритетов переходов}
    VZ:ArrayVectorOfWord; {вектор временных задержек}
    P, {количество вершин-позиций}
    D:byte; {количество вершин-переходов}
    ResultFileName:String; {имя файла результатов}
    {FlagClean:boolean; {True если сеть чистая}
    FlagTimesNet, {True если сеть временная}
    FlagPriorityNet:boolean; {True если сеть приоритетная}
    FileName:string; {имя файла результатов (без расширения)}
IMPLEMENTATION
USES KijCRTW;
CONST SetStrNumeric = ['0'..’9'];
VAR Fitext;
    P1 ,P2,D 1 ,D2,K 1 ,K2,K3:byte;
    c:char;
    FlagSawerboolean;
    FlagClean:boolean;
    TempWl ,TempW2:word;
    Temp_i,Temp_j :byte;
    TempPI: Array MatrOfW ord;
    StrProgPararn 1 :String;
procedure FatalErrores(k:byte);
begin
{if (k>4)and(k<14) then Close(F);}
case k of
1: WritelnfНе указано имя файла с матрицей прямых инциденций.');
2:Writeln('Не указано имя файла с матрицей обратных инциденций.');
3:Writelnf Не указано имя файла с вектором начальной разметки.');
4:begin
  Writelnf Ошибка чтения файла:',FileName);
  Writeln('Возможно указанный файл не существует.');
  end;
5:Writeln('OuiH6Ka строкового преобразования.');
6^гйе1п('Переполнение: слишком много столбцов в файле:',FileName);
7:begin
  Writeln('Ра3Ное количество элементов в строках матрицы,');
  Writeln(’c4HTbiBaeMoft из файла:',FileName);
  end;
8 Writelnf Файл',FileName,' пуст.');
9:Writeln(TIepenonHeHHe: слишком много строк в файле:',FileName);
10:begin
    WritelnfHecooTBeTCTBHe размерностей матриц прямых и обратных');
    Writelnf инциденций (разное количество позиций).');
    end;
11 :begin
    WritelnfHecooTBeTCTBHe размерностей матрицы прямых инциденций и');
    WritelnfBeKTopa начальной разметки (разное количество позиций).');
    end;
12:begin
    WritelnfHecooTBeTCTBHe размерностей матриц прямых и обратных');
    Writelnfинциденций (разное количество переходов).');
    end;
13:begin
    WritelnfHecooTBeTCTBHe размерностей матрицы прямых инциденций и');
    WritelnfBeKTopa приоритетов переходов (разное количество переходов).');
    end;
14:begin
    WritelnfHecooTBeTCTBHe размерностей матрицы прямых инциденций и');
    WritelnfBeKTopa временных задержек (разное количество переходов).');
    end;
else Writelnf Неизвестный сбой в программе.');
end;
Writelnf Выполнение программы прервано!');
Halt(k);
end;
procedure GetVector(var V: Array VectorOfWord; var K:byte;var flags:boolean);
var TempString:string[5];
```

```
    REsult_VAL,ITemp:integer;
    TempCharxhar;
    BTempibyte;
    BoolTemp:boolean;
begin
flags:=false;
BoolTemp:=false;
TempString:-';
BTemp:=l;
if eof(f) then FatalErrores(8);
while not BoolTemp do
begin
while not(eof(F) or eoln(F)) do
  begin
  Read(F,TempChar);
  if lOResult <> 0 then FatalErrores(4);
  if TempChar in SetStrNumeric then begin
                          TempString:=TempString + TempChar;
                          BoolTemp:=true;
                          flags:=true;
                          end
                       else
   begin
   if TempString <> ” then
     begin
     if BTemp = MAXITEM then FatalErrores(6);
     Vai (TempString, ITemp,REsult_VAL);
     if (Петр >= O)and(ITemp < 32767) then V[BTemp]:=Word(ITemp)
                            else FatalErrores(5);
     If REsult_VAL<>0 then FatalErrores(5);
     K:=BTemp;
     Inc(BTemp);
     TempString:-’;
     end;
   end;
  end;
if eoln(F)and(not BoolTemp) then Readln(F);
if eof(F) then BoolTemp:=true;
end;
if TempString <> " then
  begin
  if BTemp = MAXITEM then FatalErrores(6);
  Vai (TempString,Петр,REsult_V AL);
  if (Петр >= O)and(ITemp < 32767) then V[BTemp]:=Word(ITemp)
                          else FatalErrores(5);
  If REsult_VAL<>0 then FatalErrores(5);
  K:=BTemp;
  end;
end;
procedure GetMatr(var W: Array MatrOfWord;var X,Y:byte);
var i,k,k_old:byte;TempFlag:boolean;
begin
if eof(f) then FatalErrores(8);
i:=l;
k:=0;
while not eof(F) do
  begin
  k_old:=k;
  Get Vector (W[i] ,k ,T empFlag);
  if (k <> k_old)and(k_old <> 0) then FatalErrores(7);
  if not eof(f) then begin
                Readln(f);
                if i = MAXITEM then FatalErrores(9);
                Inc(i);
                end;
  end;
if (not TempFlag)and(eof(F) or eoln(F)) then Dec(i);
X:=k; {количество считанных столбцов}
Y:=i; {количество считанных строк}
end;
```

```
procedure ObnulVector(var V: Array VectorOfWord);
var kbyte;
begin
for i:=l to MAXITEM do V[i]:=0;
end;
procedure Control;
begin
if P1<>P2 then FatalErrores(lO);
if P1<>K1 then FatalErrores(l 1);
if D1<>D2 then FatalErrores(12);
if D1<>K2 then FatalErrores(13);
if D1<>K3 then FatalErrores(14);
end;
procedure Preview Analysis;
var i,j:byte;
   flagtemp,flgiboolean;
   tstristring;
   tstr2:string[2];
begin
{ Writelnf Проводим поиск двусторонних соединений:');
if FlagSawe then Writeln(F,'Проводим пойск двусторонних соединений:');
FlagClean:=True;              {определение чистоты сети}
{for i:=l toPdo
begin
flagtemp :=False;
forj:=ltoDdo
   begin
   if (PI[i,j]*OI[i,j] <> 0) then
     begin
     FlagClean:=False;
     Writef(p',i,' d',j,')');
      if FlagSawe then Write(F,'(p',i,' d',j,')');
     flagtemp:=True;
     end;
if flagtemp then Writelnf;');
if FlagSawe and flagtemp then Writeln(F,';');
   end;
end;
if FlagClean then begin
              WritelnfTaKHM образом данная сеть является чистой');
              if FlagSawe then Writeln(F,'Таким образом данная сеть является чистой');
              end
          else begin
              WritelnfTaKHM образом данная сеть не является чистой');
              if FlagSawe then Writeln(F,'Таким образом данная сеть не является чистой');
              end;}
FLG:=True; {анализ наличия конфликтных переходов}
WritelnfПроводим поиск конфликтующих переходов:');
if FlagSawe then Writeln(F,'Проводим поиск конфликтующих переходов:');
for i:=l to P do
begin
tstr:='';
forj:=l to D do
   begin
   Str(j:2,tstr2);
   if (PI[j,i] <> 0) then tstr:=tstr + tstr2 + '';
   end;
if Length(tstr) > 4 then
   begin
   FLG:=False;
   Writelnf Для перехода N ',i,':',tstr+';');
   if FlagSawe then Writeln(F,7IjBi перехода N ',i,':',tstr+';');
   end;
end;
if FLG then
begin
WritelnfTaKHM образом данная сеть не содержит конфликтующих переходов.');
if FlagSawe then Writeln(F,'Таким образом данная сеть не содержит конфликтующих переходов.');
end
```

```
begin
WritelnfTaKHM образом данная сеть содержит конфликтующие переходы.');
if FlagSawe then Writeln(F,Таким образом данная сеть содержит конфликтующие переходы.');
end;
end;
function ReadFileStr(var ff:text): String;
begin
end;
BEGIN
StrProgParaml:-';
if StrProgParam 1 <> " then
begin
end;
[1]№гйе1п('Укажите имя файла с матрицей прямых инциденций:');
if StrProgParaml ="then Readln(FileName)
                else begin
                     end;
IF FileName = " THEN FatalErrores(l)
            ELSE begin
                Assign(F,FileName);
                Reset(F);
                if lOResult <> 0 then FatalErrores(4);
                GetMatr(PI,Pl,Dl);
                Close(F);
                end;
\¥гйе1п('Укажите имя файла с матрицей обратных инциденций:');
if StrProgParaml = " then Readln(FileName)
                 else begin
                      end;
IF FileName = " THEN FatalErrores(2)
            ELSE begin
                Assign(F,FileName);
                Reset(F);
                if lOResult <> 0 then FatalErrores(4);
                GetMatr(OI,D2,P2);
                Close(F);
                end;
A¥гйе1п('Укажите имя файла с вектором начальной разметки:');
if StrProgParaml = " then Readln(FileName)
                 else begin
                    end;
IF FileName = " THEN FatalErrores(3)
            ELSE begin
                Assign(F, FileName);
                Reset(F);
                if lOResult <> 0 then FatalErrores(4);
                GetVector(M,K 1 ,FlagClean);
               {FlagClean использован как временная переменная
(настоящее значение этого флага определяется в PreviewAnalysis;)}
                Close(F);
                end;
[1]\¥гйе1п('Укажите имя файла с вектором приоритетов переходов:');
if StrProgParaml = " then Readln(FileName)
                else begin
                     end;
IF FileName = " THEN begin
                Obnul Vector (VP);
                K2:=D1;
                FlagPriorityN et :=False;
                end
            ELSE begin
                Assign(F, FileName);
                Reset(F);
                if lOResult <> 0 then FatalErrores(4);
                GetVector(VP,K2, FlagClean);
                Close(F);
                FlagPriorityNet:=True;
                end;
Writeln('Укажите имя файла с вектором временных задержек:');
```

```
                 else begin
                      end;
IF FileName = '' THEN begin
                 ObnulVector(VZ);
                 K3:=D1;
                 FlagTimesNet:=False;
                 end
              ELSE begin
                 Assign(F,FileName);
                 Reset(F);
                 if lOResult <> 0 then FatalErrores(4);
                 GetVector(VZ,K3,FlagClean);
                 Close(F);
                 FlagTimesNet: =True;
                 end;
Control;
D:=D1;
P:=P1;
Writein;
repeat
Writelnf Введите имя файла результатов ( если уже существует то будет');
Write('ЗаМеНеН), без расширения:');
if StrProgParam 1 =”then Readln(ResultFileName)
                 else begin
                      end;
FileName:=ResultFileName;
Writeln('Исходные данные загружены.');
IF FileName = '' THEN Writeln('Ввод пустой строки недопустим!');
until FileName <> '';
{while Keypressed do c:=ReadKey;}
Writef Сохранять файл предварительного анализа?');
repeat
{if keypressed then c:=ReadKey;}
ReadLn(c);
until (c-Y')or(c='y')or(c='N')or(c='n');
{Writeln(c); }
if (c='Y')or(c='y') then FlagSawe:=True
                 else FlagSawe:=False;
if FlagSawe then begin
              Assign(F,ResultFileName + '.TXT');
              Rewrite(F);
              if lOResult <> 0 then begin
                Writelnf Ошибка создания файла:',ResultFileName + '.TXT');
                Halt(15);
                              end;
              end;
Preview Analysis;
if FlagSawe then
begin
if FlagTimesNet then Writeln(F,'CeTb является временной.')
              else Writeln(F,'CeTb не является временной.');
end;
if FlagTimesNet then WritelnfCeTb является временной.')
             else WritelnfCeTb не является временной.');
if FlagSawe then
begin
if FlagPriorityNet then Writeln(F,’CeTb является приоритетной.')
                 else Writeln(F,'CeTb не является приоритетной.');
end;
if FlagPriorityNet then WritelnfCeTb является приоритетной.')
               else WritelnfCeTb не является приоритетной.');
if FlagSawe then Close(F);
Writelnf Предварительный анализ завершен.');
{для удобства дальнейшего использования транспонируем матрицу PI}
TempPI:=PI;
for Temp_i:=l to Maxitem do
for Temp_j:=l to Maxitem do
   begin
   PI[Temp_i,TempJ]:=TempPI|TempJ,Temp_i];
   end;
```

```
Writeln(7lnH продолжения нажмите Enter.');
Readln;
END.
```

```
UNIT Typetree;
INTERFACE
CONST MAXITEM = 85; {не более 85 !!!!!}
     CvantltemMax = 127; {не более 127 !!!!!}
TYPE
    Array VectorOfWord = array[1..MAXITEM] of word;
    ArrayMatrOfWord = array[1..MAXITEM] of Array VectorOfWord;
    TTimeOutVar = array[0..MAXITEM,0..CvantltemMax] of shortint; {таблица задержек}
    TInfoNumeric = array[0..MAXITEM] of byte; {вектор данных разного типа}
    TTableStimulationPassage = arrayfl ..MAXITEM] of byte; {таблица возбужденных переходов}
    TStage = Record {Сценарии}
          Size:byte;
          V:array[1..255] of TInfoNumeric;
          end;
    TTypeltem = (Internal,Sheet,W_Sheet,BluntKnife,Repetition,Unknown); {тип вершины (внутренняя (Internal),
nncT(Sheet), W-nHCT(W_Sheett),TynHK(BluntKnife),noBTop(Repetition),HeH3BecTHo(Unknown))}
    TItemTree = record
             NU:longint; {уровень(такт)}
             M:ArrayVectorOfWord; {разметка}
             P:TInfoNumeric; {номера сработавших переходов}
             TimeTablezTTimeOutVar; {Таблица задержек меток в переходах}
             OldPoint_ID:longint; {ссылка на предшественника(родителя)}
             TypeItem:TTypeItem; {тип вершины}
             FlagNew:boolean; {True если вершина исследована}
                      W_Point_ID:longint; {ссылка по ID (если вершина не внутренняя,иначе = -1)}
             Point_ID_NE:longint; {ссылка на предыдущий элемент данного уровня(-1 если нет) с тем же
                                   родителем}
             Point_ID_FirstDescendant:longint; {ссылка на первого потомка}
             Point_ID_LastDescendant:longint; {ссылка на последнего потомка}
             NumberDescendantrlongint; {число потомков (необязательное поле)}
             end;
IMPLEMENTATION
END.
```

```
UNIT TimeOut;
INTERFACE
USES TypeTree;
TYPE PTimeOutVar = ^TTimeOutVar;
VAR TSM,Old_TSM:PTimeOutVar; {таблица поглощенных меток [переход;задержка]}

procedure NewTakt(var TSP:TTimeOutVar); {новый такт}
procedure SwallowingAndDeliveranceMalk( {поглощение и освобождение меток}
                              P, {кол-во позиций}
                              D:byte; {кол-во переходов}
                              const MPI:ArrayMatrOfWord;{МаТр. прямых инцинденций}
                              const MOI:ArrayMatrOfWord; {матр. обратных инцинденций}
                              const NPassage:TInfoNumeric; {номера сработавших переходов}
                              var TSP:TTimeOutVar; {таблица состояний переходов}
                              var VR:ArrayVectorOfWord; {вектор разметки}
                              const VZ:ArrayVectorOfWord; {вектор задержек}
                              FlagTimeNet:boolean);{True если сеть временная}
IMPLEMENTATION
VAR PExitProcTO:pointer;
procedure ObnulTimeOutVar(var TSP:TTimeOutVar);
var i,j:byte;
begin
for i:=0 to MAXITEM do
   for j:=0 to CvantltemMax do TSP[i,j]:=-1;
end;
procedure NewTakt(var TSP:TTimeOutVar);
var i,j: byte;
begin
for i:=l to MAXITEM do
   for j:=0 to CvantltemMax do
    if TSP[ij] > 0 then Dec(TSP[i,j]);
```

```
end;
procedure SwallowingAndDeliveranceMalk( {поглощение и освобождение меток}
                                P, {кол-во позиций}
                                D:byte; {кол-во переходов}
                                const MPI:ArrayMatrOfWord;{МаТр. прямых инцинденций}
                                const MOI:ArrayMatrOfWord; {матрица обратных инцинденций}
                                const NPassage:TInfoNumeric; {номера сработавших переходов}
                                var TSP:TTimeOutVar; {таблица состояний переходов}
                                var VR:ArrayVectorOfWord; {вектор разметки}
const VZ:ArrayVectorOfWord; {вектор задержек}
FlagTimeNet:boolean);{True если сеть временная}
var i,j ,k:byte;flg:boolean;
begin
{поглощение меток сработавшими переходами}
for i:= 1 to NPassagefO] do
   begin
   flg:=true;
   k:=0;
   while (fig) do
     begin
     if TSP[NPassage[i],k] < 0 then begin
                          TSP[NPassage[i],k]:=VZ[NPassage[i]];
                          flg:=false;
                          end;
     if k < CvantltemMax then Inc(k)
                  else begin
                      Writeln('Переполнение таблицы состояний переходов.');
                      Halt(54);
                      end;
     end;
    for j:=l to P do VR[j] := VR|j] - MPIQ,NPassage[i]]; {обновление разметки}
   end;
{освобождение меток с нулевой задержкой всеми переходами}
for i:=l to D do
   for j:=0 to CvantltemMax do
     if TSP[i,j] = 0 then
       begin
       for k:=l to P do VR[k] := VR[k] + MOI[k,i]; {обновление разметки}
       TSP[i,j]:=-l;
       end;
NewTakt(TSP);
end;
procedure TimeOutDestuct;far;
begin
exitproc:=PExitProcTO;
Dispose(TSM);
Dispose(Old_TS M);
end;
BEGIN
TSM:=nil;
Old_TSM:=nil;
New(TSM);
New(Old_TSM);
ObnulTimeOut V ar(TS M ^A);
ObnulTimeOutVar(Old_TSM^A);
PExitProcTO:=exitproc;
exitproc:=@TimeOutDestuct;
```

```
UNIT Units;
INTERFACE
USES TypeTree;
TYPE PStimulationPassage=^ATTableStimulationPassage;
    PStage=^ATStage;
VAR StimulationPassage:PStimulationPassage;
    Stage:PStage; {поиск возбужденных переходов}
procedure FindStimulationPassage(P,D:byte;
            const MPLArrayMatrOfWord; {матрица прямых инцинденций}
            const VM:ArrayVectorOfWord; {вектор разметки}
            var Result :TT ableStimulationPassage);
```

```
                {поиск сценариев срабатывания}
  procedure FindStage(flagprior:boolean; {флаг приоритетности сети}
                  const TVP:TTableStimulationPassage;{Та6n. возбужд.переходов}
                  const VTR:ArrayVectorOfWord; {вектор текущей разметки}
                  const VPP:ArrayVectorOfWord; {вектор приоритетов переходов}
                  const MPI:ArrayMatrOfWord; {матр. прям, инцинденций}
                  const P,D:byte;              {размерность MPI}
                  var Result:TStage); {результат}
  function ComparisonItemTree( {сравнение вершин}
        const Rl:ArrayVectorOfWord; {разметка}
        const TSP1 :TTimeOutVar; {таблица состояний переходов}
        const R2:ArrayVectorOfWord; {разметка}
        const TSP2:TTimeOutVar; {таблица состояний переходов}
        const MOI:ArrayMatrOfWord; {матрица обратных инцинденций}
        Flag:boolean; {True если временная сеть(с задержками)}
        P, {количество позиций}
        D:byte {количество переходов}
        ):char; {сравнение разметок ,
  возвращает (относительно первых двух параметров) >,? или = }
  procedure Indicate(D:longint); {индикатор выполнения}
IMPLEMENTATION
USES KijCRTW;
VAR PExitProc.-pointer;

  procedure Units lDestuct;far;
  begin
  exitproc:=PExitProc;
  Dispose(StimulationPassage);
  Dispose(Stage);
  end;
{=====================================================================}
  procedure FindStimulationPassage(P,D:byte;
            const MPI:ArrayMatrOfWord;
            const VM:ArrayVectorOfWord;
            var Result:TTableStimulationPassage);
  var i,j:byte;
     flg:boolean;
  begin
  for j:=l to d do
    begin
    flg:=True;
    for i:=l to p do if MPI[i,j] > VM[i] then flg:=False;
    if fig then Result[j]:=255
         else Result[j]:=0;
    end;
  end;

  procedure FindStage(flagprior:boolean;
                  const TVP:TTableStimulationPassage;
                  const VTR:ArrayVectorOfWord;
                  const VPP:ArrayVectorOfWord;
                  const MPI:ArrayMatrOfWord;
                  const P,D:byte;
                  var Result:TStage);
  type TTemp = array[1..MAXITEM] of byte;
       PTemp = ^ TTemp;
  var Temp_i,i,j,k,l,ll,q,w:byte;
      temp integer;
      TempSize,TempSizel :byte;
      fig,flag,FStop,TempFlag,TF,FlagDel:boolean;
      TPRP,TempVP:TInfoNumeric;
      Temp VR: Array VectorOfW ord;
      Point:PTemp;
  begin
  New(Point);
  Result.Size:=O;
  Temp:=255;
  FStop:=True;
  while (Temp >= O)or(FStop) do
  begin
```

```
FStop:=False;
TPRP[O]:=O;
if flagprior then {выявленное возб. группы переходов с равными приоритетами}
   begin
   repeat
   for i:=l to D do
    begin
    if (TVP[i]=255)and(VPP[i]=temp) then
     begin
     Inc(TPRP[0]);
     TPRP[TPRP[O]]:=i;
     end;
    end;
   if TPRP[0]=0 then Dec(temp);
   until (TPRP[0]>0)or(temp<0);
   end else
    begin
     for i:= 1 to D do if TVP[i]=255 then
     begin
     Inc(TPRP[0]);
     TPRP[TPRP[O]]:=i;
     end;
    temp:=0;
    end;
if Result.Size = 0 then for i:= 1 to TPRP[0] do {инициализация таблицы сценариев}
   begin
   Inc(Result.Size);
   Result. V[Result. Size, 0]:=1;
   Result.V[Result.Size,l]:=TPRP[i];
   FStop:=True;
   end else
   begin
   Flag:=False;
   while not Flag do
    begin
    Flag:=True;
    T empSize:=Result. Size;
    for Temp_i:=l to TempSize do
     begin
     {определеним для строки таблицы сценариев получаемую разметку}
     TempVR:=VTR;
     for j:=l to Result.V[Temp_i,0] do
      fork:=ltoPdo
      begin
      if TempVRfk] < MPI[k,Result. V|Temp_i,j]] then
            begin
            \¥гйе1п('Критическая ошибка в программе!');
            end;
      Temp VR[k]:=TempVR[k]-MPI[k,Result.V[TempJ,j]];
      end;
   {определим для такой разметки возбужденные переходы,входящие в группу равноприоритетных}
     TempVP[0]:=0;
     for j:=l to d do
      begin
      flg:=True;
      for i:=l to p do if MPI[i,j] > TempVR[i] then flg:=False;
      TempFlag:=False;
      for 1:= 1 to TPRP[0] do if TPRPfl] = j then TempFlag:=True;
      for 1:=1 to Result.V[Temp_i,0] do
       begin
       if Result. V[Temp_i,l] =j then TempFlag:=False;
       end; {for}
      if (flg)and(TempFlag) then begin
                  Inc(TempVP[O]);
                  T emp VP|T emp VP[0]]: =j;
                  end;
      end;
   {расширим таблицу сценариев}
     if (TempVP[0] <> 0)and(not (Result.V[Temp_i,0] = 0)) then
```

```
        for 1:=1 to TempVPfO] do
         begin
         if not(Result.Size < 255) then
          begin
           Writeln('nepenonHeHHe таблицы сценариев.');
           Halt(50);
          end;
         Inc(Result.Size);
         Result. V[Result.Size]:=Result.V[Temp_i];
         Inc(Result. VfResul t. Size, 0]);
         Result. VfResult.Size, Result. V[Result.Size,O]]:=TempVP[l];
         end;
       Flag:=False;
       FStop:=True;
       Result. V(Temp_i,0]:=0;
       end;
   {поиск одинаковых(логически) строк и обнуление лишних}
      for q:= 1 to Result.Size do
       begin
       for w:=l to MAXITEM do Point A[w]:=0;
       for w:=l to Result.V[q,0] do Point A[Result.V[q,w]]:=255;
       for ll:=q+1 to Result.Size do
         begin
         FlagDel:=True;
         for w:=l to Result. V[ll,0] do
          if PointA[Result.V[ll,w]] <> 255 then FlagDek-False;
          if FlagDel then Result.V[ll,0]:=0;
         end;
       end;
    {уничтожение пустых строк в таблице сценариев}
       T empSize 1:=Result. Size;
      for ll:=TempSizel downto 1 do
       begin
       if Result. V[ll,0] = 0 then
         begin
          for q:= 11 to Result.Size-1 do
           begin
           Result. V[q]:=Resul t. V[q+1 ];
           end;
          Dec(Result.Size);
         end;
       end;
      end;
    end;
   end;
if (temp > -l)and(not FStop) then Dec(Temp);
end;
Dispose(Point);
end;
[==================================»=======================)
function ComparisonItemTree( {сравнение вершин}
         const Rl:ArrayVectorOfWord; {разметка}
         const TSPlrTTimeOutVar; {таблица состояний переходов}
         const R2:ArrayVectorOfWord; {разметка}
         const TSP2:TTimeOutVar; {таблица состояний переходов}
         const MOEArrayMatrOfWord; {матрица обратных инцинденций}
         Flagiboolean; {True если временная сеть(с задержками)}
         P, {количество позиций}
         D:byte {количество переходов}
         ):char; {сравнение разметок,
       возвращает (относительно первых двух параметров) >,? или = }
var C,C1 ,TempResult 1 ,TempResult2:char;
     i,j,k:byte;
     TI,T2:array[l..MAXITEM,0..CvantltemMax] of byte; {табл, задержек}
     Pl,P2:array[l..MAXITEM,0..CvantltemMax] of 1o^iЩ;{табл. поступлений меток в позиции}
begin
{сравнение разметок}
C:='=';
for i:=l to P do
   begin
```

```
  if R1 [i] >= R2[i] then
    begin
    if R1 [i] > R2[i] then if C <> '?' then C:='>';
    end else C
end;
{сравнение таблиц состояния переходов}
if Flag then
begin
TempResult 1
TempResult2:='=';
FillChar(Tl,SizeOf(Tl),0);
FillChar(T2,SizeOf(T2),0);
for i:= 1 to D do
    begin
    for j:=0 to CvantltemMax do
      begin
      if TSPl[i,j] >= 0 then
                  begin
                    Tl[i,TSPl[i,j]]:=l;
                  end;
      if TSP2[i,j] >= 0 then
                  begin
                    T2[i,TSP2[iJ]]:= 1;
                  end;
                end;
            end;
FillChar(Pl,SizeOf(Pl),0);
FillChar(P2,SizeOf(P2),0);
for k:=0 to CvantltemMax do
    begin
    for i:=l to P do
      begin
      for j:=l to D do
        begin
        Pl[i,k]:=Pl[i,k]+(Tl[j,k]*MOI[i,j]);
        P2[i,k]:=P2[i,k]+(T2[j,k]*MOI[i,j]);
        end;
      end;
    end;
for i:=l to P do
    begin
    for j:=0 to CvantltemMax do
      begin
      if Pl [i,j] < P2[i,j] then TempResult 1
      if Pl[ij] <> P2[i,j] then TempResult 2:-?';
      end;
    end;
if TempResult2 = '=' then C1
                  else C1 :=TempResult 1;
end else Cl
{нахождение общего результата}
if (C->')and(Cl->') then ComparisonItemTree:='>';
if (C='>')and(Cl='=') then ComparisonltemTree:->';
if (C->')and(Cl-?') then ComparisonltemTree:-?';
if (C-=')and(Cl->') then ComparisonltemTree:->';
if (C--)and(Cl=-') then ComparisonltemTree:- -;
if (C= -')and(Cl-?') then ComparisonltemTree:-?';
if (C-?')and(C !='>') then ComparisonltemTree:='?';
if (C-?')and(Cl= -') then ComparisonltemTree:-?';
if (C=*?')and(d =*?*) then ComparisonltemTree:='?';
end;
{================================================================}
procedure Indicate(D:longint); {индикатор выполнения}
begin
Go t oX Y(l, Where Y);
Write('ripoc4HTaHo: ',D,' вершин.');
end;

BEGIN
StimulationPassage:=nil;
```

```
Stage:=nil;
N ew(StimulationPassage);
New(Stage);
PExitProc:=exitproc;
exitproc:=@Units 1 Destuct;
END.
```

UNITIOTree;

```
INTERFACE
USES TypeTree,MStream;
procedure InitialiseFile(Name:string); {инициализация}
procedure DeinitialiseFile; {деинициализация}
procedure MakeUpItemTree(var ItemTreeiTItemTree); {добавление элемента списка}
procedure WriteItemTree(var ItemTree:TItemTree;ID:longint); {запись элемента списка}
procedure ReadItemTree(var ItemTree:TItemTree;ID:longint); {чтение элемента списка}
function Search_WList(const ItemTree:TItemTree;
                const ID:longint;
                const MOI:ArrayMatrOfWord;
                var TypeItem:TTypeItem;
                P, D:byte;
                FlagTempNet:boolean):longint; {поиск w-листов, листов и повторов}
function FindNewExploreItemTree( {поиск новой исследуемой вершины (-1 если не найдена)}
                IDdongint {номер элемента с которого начнется поиск}
                  ):longint; {номер найденого элемента или -1}
function IDFlowingItemTree:longint;
IMPLEMENTATION
USES Units;
VAR F:file of TItemTree;
  FlagMemoryStream:boolean;
  c:char;
procedure InitialiseFile(Name:string);
begin
if FlagMemoryStream then MInitialiseFile(Name)
else begin
Assign(F,Name + '.TR');
Rewrite(F);
if lOResult <> 0 then begin
                WritelnfOinnGKa создания файла результата!');
                Halt(16);
                end;
Close(F);
Reset (F);
if lOResult <> 0 then begin
                Writeln('OniH6Ka открытия файла результата!');
                Halt(17);
                end;
end; {else}
end;
procedure MakeUpItemTree(var ItemTree:TItemTree);
begin
if FlagMemoryStream then MMakeUpItemTree(ItemTree)
else begin
  Seek(F,FileSize(F));
  Write(F,ItemTree);
end; {else}
end;
procedure WriteItemTree(var ItemTree:TItemTree;ID:longint);
begin
if FlagMemoryStream then MWriteltemTree(ItemTreeJD)
else begin
  Seek(F,ID);
  Write(FJtemTree);
end; {else}
end;
procedure ReadItemTree(var ItemTree:TItemTree;ID:longint);
begin
if FlagMemoryStream then MReadItemTree(ItemTree,ID)
else begin
  Seek(FJD);
  Read(FJtemTree);
```

```
end; {else}
end;
function Search_WList(const ItemTreerTItemTree;
                const IDdongint;
                const MOI:ArrayMatrOfWord;
                var TypeItem:TTypeItem;
                P,
                D:byte;
                FlagTempNet:boolean):longint; {поиск w-листов,листов и повторов}
var TempIDdongint;
   PTempItemTree:TItemTree;
   TempFlag:boolean;
   TempResultType.'TTypeItem;
   TempResult ID dongint;
begin
TempFlag:=true;
TempID:=ItemTree.OldPoint_ID;
TempResultType:=Internal;
TempResultID:=-1;
while (TempID >= 0)and Tempflag do
  begin
  if FlagMemoryStream then MReadItemTree(PTempItemTree,TempID)
                else ReadItemTree(PTempItemTree,TempID);
   case
ComparisonItemTree(ItemTree.M,ItemTree.TimeTable,PTempItemTree.M,PTempItemTree.TimeTable,MOI,FlagTe
mpNet,P,D) of
   '>':begin
      TempFlag:=false;
      TempResultType:=W_Sheet;
      if FlagMemoryStream then TempResultID:=MFilePos - 1
                else TempResultID:=FilePos(F) - 1;
      end;
   '=':begin
      TempFlag:=false;
      T empResul tT ype:=Sheet;
      if FlagMemoryStream then TempResultID:=MFilePos - 1
                else TempResultID:=FilePos(F) - 1;
      end;
   end;
  TempID:=PTempItemTree.OldPoint_ID;
  end;
{поиск повторов}
if TempResultType = Internal then
  begin
  TempFlag:=True;
  TempID:=0;
  if FlagMemoryStream then MSeek(2)
                else Seek(F,2);
  if FlagMemoryStream then
  begin
  while (MFilePos < ID)and Tempflag do
   begin
   MRead(PTempItemTree);
     case
ComparisonItemTree(ItemTree.M,ItemTree.TimeTable,PTempItemTree.M,PTempItemTree.TimeTable,MOI,FlagTe
mpNet,P,D) of
     '=':begin
        TempFlag:=false;
        TempResultType:=Repetition;
        TempResultID:=MFilePos - 1;
        end;
     end;
   end;
  end else
  begin
  while (FilePos(F) < ID)and Tempflag do
   begin
   Read(F,PTempItemTree);
```

```
      case
ComparisonItemTree(ItemTree.M,ItemTree.TimeTable,PTempItemTree.M,PTempItemTree.TimeTable,MOI,FlagTe
mpNet,P,D) of
      -'•.begin
         TempFlag:=false;
         TempResultType:=Repetition;
         TempResultID:=FilePos(F) - 1;
         end;
      end;
     end;
   end; {else}
  end;
Typeitem: =TempResultType;
Search_WList: =T empResult ID;
end;
function IDFlowingItemTree:longint;
begin
if FlagMemoryStream
  then IDFlowingItemTree:=MIDFlowingItemTree
  else IDFlowingItemTree:=FilePos(F);
end;
procedure DeinitialiseFile;
begin
if FlagMemoryStream then MDeinitialiseFile
                else Close(F);
end;
function FindNewExploreItemTree( {поиск новой исследуемой вершины (-1 если не найдена)}
                          ID:longint {номер элемента с которого начнется поиск}
                          ):longint; {номер найденого элемента или -1}
var Temp:^TItemTree;
    TempFlag:boolean;
begin
if FlagMemoryStream then
begin
New(Temp);
MSeek(ID);
T empFlag:=true;
while                                (not                                MEof)and(TempFlag)                                do
  begin
  MRead(Temp^);
  if not Temp ^ .FlagNew then begin
                    FindNewExploreltemTree := MFilePos - 1;
                     TempFlag:=false;
                    end;
  end;
if TempFlag then FindNewExploreltemTree := -1;
Dispose(Temp);
end else
begin
NewfTemp);
Seek(FJD);
TempFlag:=true;
while                                (not                                Eof(F))and(TempFlag)                                do
  begin
  Read(F,Temp^);
  if not Temp^.FlagNew then begin
                    FindNewExploreltemTree := FilePos(F) - 1;
                     TempFlag:=false;
                    end;
  end;
if TempFlag then FindNewExploreltemTree := -1;
DisposefTemp);
end; {else}
end;
BEGIN
Writeln('CTpoHTb дерево: M - в памяти с последующим сохранением,');
Write('D - в файле: ');
repeat
ReadLn(c);
```

```
if (c='M')or(c='m') then FlagMemoryStream:=True
                    else FlagMemoryStream:=False;
END.
```

```
UNIT MStream; {аналог ЮТree при построении дерева в памяти}
interface
uses TypeTree;
procedure MInitialiseFile(Name:string); {инициализация}
procedure MDeinitialiseFile; {деинициализация}
procedure MMakeUpItemTree(var ItemTreerTItemTree);
procedure MWriteItemTree(var ItemTree:TItemTree;ID:longint); {запись элемента списка}
procedure MReadItemTree(var ItemTree:TItemTree;ID:longint); {чтение элемента списка}
function MIDFlowingltemTreerlongint;
procedure MRead(var ItemTreerTItemTree);
function MEofrboolean;
function MFilePosrlongint;
procedure MSeek(ID:longint);
implementation
uses Classes;
var ResultFileName:String;
    MemStream:TMemoryStream;
    Pos, ItemSize, TempPos,LastPos:longint;
procedure MInitialiseFile(Name:string); {инициализация}
begin
Result FileN ame:=N ame;
MemStream:=TMemoryStream.Create;
Pos:=0;
TempPos:=0;
LastPos:=0;
ItemSizer=SizeOf(TItemTree);
MemStream.Sizer=1000*SizeOf(TItemTree);
end;
procedure MDeinitialiseFile;
begin
MemStream.Size:=LastPos;
Writeln('Производится сохранение результатов в файле...');
MemStream.SaveToFile(ResultFileName + '.TR');
MemStream.Destroy;
end;
procedure MMakeUpItemTree(var ItemTreerTItemTree);
begin
MemStream.Seek(LastPos,soFromBeginning);
MemStream.WriteBuffer(ItemTree, ItemSize);
Last Pos:=MemStream .Position;
if MemStream.Size = LastPos
then MemStream.Size := MemStream.Size + 1000*SizeOf(TItemTree);
end;
procedure MWriteItemTree(var ItemTree:TItemTree;ID:longint); {запись элемента списка}
begin
MemStream.Seek(ID*ItemSize,soFromBeginning);
MemStream.WriteBuffer(ItemTreeJtemSize);
if MemStream.Position > LastPos then LastPos := MemStream.Position;
end;
procedure MReadItemTree(var ItemTreerTItemTree;IDrlongint); {чтение элемента списка}
begin
MemStream.Seek(ID*ItemSize,soFromBeginning);
MemStream.ReadBuffer(ItemTree,ItemSize);
end;
function MIDFlowingltemTreerlongint;
begin
Resultr=(MemStream.Position) div ItemSize;
if Result*sizeOf(TItemTree) <> MemStream.Position then
begin
Writeln('OuiH6Ka распределения памяти. Программа аварийно завершена.');
halt(55);
end;
end;
procedure MRead(var ItemTreerTItemTree);
begin
TempPosrr(MemStream.Position) div ItemSize;
```

```
MemStream.Seek(TempPos*ItemSize,soFromBeginning);
MemStream.ReadBuffer(ItemTree,ItemSize);
TempPos:=(MemStream.Position) div ItemSize;
end;
function MEof:boolean;
begin
if MemStream.Position = LastPos
then Result:=True
else Result:=False;
end;
function MFilePos:longint;
begin
Result:=MemStream.Position div ItemSize;;
end;
procedure MSeek(ID-.longint);
begin
MemStream.Seek(ID*ItemSize,soFromBeginning);
end;
end.
```

Листинг программы **“Explorer”,** которая использует стандартные элементы Windows и Delphi ограничен только модулем основного файла (EXPLORER.DPR), модуля, реализующего чтение дерева (iotree.pas) и модуля с описанием типа элементов дерева (typetree.pas).

```
program Projectl; {Explorer}
uses
  Forms,
  Unitl in ’Unit 1.pas’ {Explorer},
  Unit2 in ’Unit2.pas' {Form2},
  Unit3 in ’Uhit3.pas’ {Form3},
  Unit4 in ’Unit4.pas’,
  Unit5 in ’Unit5.pas’ {Form5},
  Unit6 in ’Unit6.pas’ {FindM},
  Unit? in ’Unit?.pas' {ResultFind},
  Unit8 in ’UnitS.pas’ {FindID},
  Unit9 in ’Unit9.pas’ {Form9};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TExplorer, Explorer);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm5, Form5);
  Application.CreateForm(TFindM, FindM);
  Application.CreateForm(TResultFind, ResultFind);
  Application.CreateForm(TFindID, FindID);
  Application.CreateForm(TForm9, Form9);
  Application.Run;
end.
```

```
UNITIOTree;
INTERFACE
USES TypeTree;
procedure InitialiseFile(Name:string); {инициализация}
procedure DeinitialiseFile; {деинициализация}
procedure MakeUpItemTree(var ItemTreeiTItemTree); {добавление элемента списка}
procedure WriteItemTree(var ItemTree:TItemTree;ID:longint); {запись элемента списка}
```

```
procedure ReadItemTree(var ItemTree:TItemTree;ID:longint); {чтение элемента списка}
function Search__WList(const ItemTree:TItemTree;
                    const IDdongint;
                    var TypeItem:TTypeItem;
                     FlagTempNet:boolean)dongint; {поиск w-листов,листов и повторов}
function FindNewExploreItemTree( {поиск новой исследуемой вершины (-1 если не найдена)}
ID:longint {номер элемента с которого начнется поиск}
                        ):longint; {номер найденого элемента или -1}

IMPLEMENTATION
USES Units;
VAR F:file of TItemTree;

procedure InitialiseFile(Name:string);
begin
Assign(F,Name + '.TR');
Rewrite(F);

if lOResult <> 0 then begin
                    Writeln('0mn6Ka создания файла результата!');
                    Halt(16);
                    end;
Close(F);
Reset(F);
if lOResult <> 0 then begin
                    \Угйе1п('Ошибка открытия файла результата!');
                    Halt(17);
                    end;

end;

procedure MakeUpItemTree(var ItemTree:TItemTree);
begin
   Seek(F,FileSize(F));
   Write(FJtemTree);
end;

procedure WriteItemTree(var ItemTree:TItemTree;IDdongint);
begin
   Seek(F,ID);
   Write(FJtemTree);
end;

procedure ReadItemTree(var ItemTree:TItemTree;ID:longint);
begin
   Seek(FJD);
   Read(FJtemTree);
end;

function Search_WList(const ItemTree:TItemTree;
                    const IDdongint;
                    var TypeItem:TTypeItem;
                     FlagTempNet:boolean):longint; {поиск w-листов,листов и повторов}
var TempIDdongint;
     PTempItemTree:^TItemTree;
     TempFlag:boolean;
     TempResultType:TTypeItem;
     T empResul t ID: longint;
begin
New(PTempItemTree);
{поиск листов(=) и W-листов (>=)}
TempFlag:=true;
TempID:=ItemTree.OldPoint_ID;
TempResultType:=Internal;
TempResultID:=-l;
while (TempID >= O)and Tempflag do
   begin
```

```
  ReadItemTree(PTempItemTree^A,TempID);
  case
ComparisonItemTree(ItemTree.M,ItemTree.TimeTable,PTempItemTree^A.M,PTempItemTree^A.TimeTable,Flag
TempNet) of
  '>':begin
      TempFlag:=false;
      TempResultType:=W_Sheet;
      TempResultID:=FilePos(F) - 1;
      end;
  '=':begin
      TempFlag:=false;
      TempResultType:=Sheet;
      TempResultID:=FilePos(F) - 1;
      end;
  end;
  TempID:=PTempItemTree^A.OldPoint_ID;
  end;
{поиск повторов}
if TempResultType = Internal then
  begin
  TempFlag:=True;
  TempID:=0;
  while (TempID < ID)and Tempflag do
    begin
    ReadItemTree(PTempItemTree^A,TempID);
    case
ComparisonItemTree(ItemTree.M,ItemTree.TimeTable,PTempItemTree^A.M,PTempItemTree^A.TimeTable,Flag
TempNet) of
    '=':begin
        TempFlag:=false;
        TempResultType:=Repetition;
        TempResultID:=FilePos(F) - 1;
        end;
    end;
    end;
  end;
Dispose(PTempItemTree);
TypeItem:=TempResultType;
Search_WList:=TempResultID;
end;

function IDFlowingItemTreedongint;
begin
  IDFlowingItemTree:=FilePos(F);
end;

procedure DeinitialiseFile;
begin
Close(F);
end;

function FindNewExploreItemTree( {поиск новой исследуемой вершины (-1 если не найдена)}
                        IDdongint {номер элемента с которого начнется поиск}
                        ):longint; {номер найденого элемента или -1}
var Temp:^ATItemTree;
    TempFlag:boolean;
begin
New(Temp);
Seek(FJD);
TempFlag:=true;
while (not Eof(F))and(TempFlag) do
  begin
  Read(F,Temp^A);
  if not Temp^A.FlagNew then begin
                    FindNewExploreItemTree := FilePos(F) - 1;
                    TempFlag:=false;                    end;
   end;
```

```
if TempFlag then FindNewExploreltemTree := -1;
Dispose(Temp);
end;

BEGIN
END.
```

UNIT Typetree;

```
INTERFACE
CONST MAXITEM = 85; {не более 85 !!!!!}
      CvantltemMax = 127; {не более 127 !!!!!}
TYPE
    Array VectorOfWord = array[l..MAXITEM] of word;

    ArrayMatrOfWord = array[1..MAXITEM] of ArrayVectorOfWord;

    TTimeOutVar = array[0..MAXITEM,0..CvantltemMax] of shortint; {таблица задержек}

    TInfoNumeric = array[0..MAXITEM] of byte; {вектор данных разного типа}

    TTableStimulationPassage = array[1..MAXITEM] of byte;
                        {таблица возбужденных переходов}

    TStage = Record {Сценарии}
           Size:byte;
           V:array[1..255] of TInfoNumeric;
           end;

     TTypeltem = (Internal,Sheet,W_Sheet,BluntKnife,Repetition,Unknown); {тип вершины
           (BHyTpeHHnn(Internal),nHCT(Sheet),W-
лист(W_Sheett),тупик(BluntKnife),повтор(Repetition),неизвестно(Unknown))}


    TItemTree = record
             NU:longint; {уровень(такт)}
             M:ArrayVectorOfWord; {разметка}
             P:TInfoNumeric; {номера сработавших переходов}
             TimeTable.TTimeOutVar; {Таблица задержек меток в переходах}
             OldPoint_ID:longint; {ссылка на предшественника(родителя)}
             TypeltemzTTypeltem; {тип вершины}
             FlagNew:boolean; {True если вершина исследована}
             W_Point_ID:longint; {ссылка по ID (если вершина не внутренняя,иначе = -1)}
             Point_ID_NE:longint; {ссылка на предыдущий элемент данного уровня(-1 если нет) с тем же
родителем}
             Point_ID_FirstDescendant:longint; {ссылка на первого потомка}
             Point_ID__LastDescendant:longint; {ссылка на последнего потомка}
             NumberDescendantdongint; {число потомков (необязательное поле)}
             end;

IMPLEMENTATION
END.
```

ПРИЛОЖЕНИЕ Б

Программа построения матриц инцидентности по чертежу модели

Блок-схема алгоритма программы представлена на рис Б.1. Для написания ее использован язык программирования AutoLISP, который встроен в графический редактор AutoCAD. Ниже следуют пояснения ней.

Начало. Присвоение переменным **(пп** и **Inn)** набора, состоящего из всех элементов чертежа, и его длины; “очистка” используемых в дальнейшем переменных.

Создание послойных наборов. Выделяется слой **1ay0,** содержащий переходы подсети алгоритма (в примере — “1”). В отдельных переменных со сконструированными именами вида **nabj** (j — № набора) формируются наборы из элементов выделенного слоя и одного из других обнаруженных слоев. Операция повторяется до окончания всех существующих слоев.

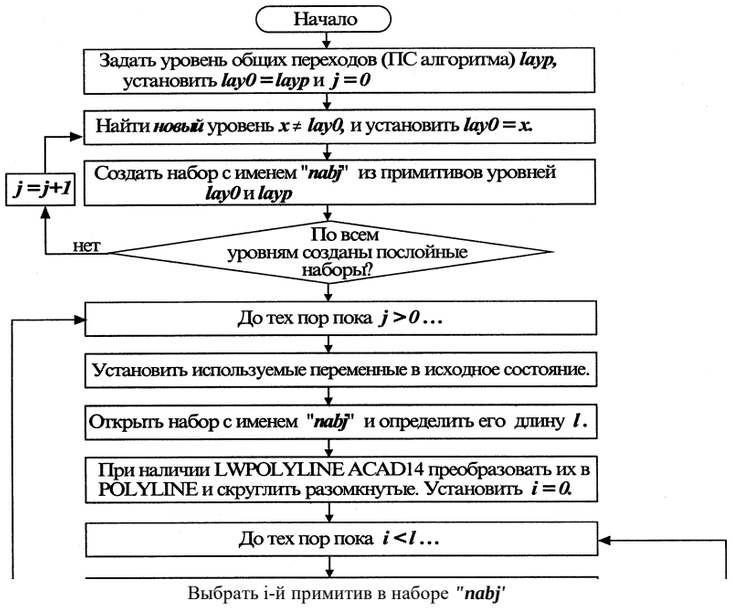
Ранжировка примитивов в послойных наборах. В цикле по j конвертируются примитивы типа LWPOLYLINE (для совместимости с чертежами, созданными в более ранних версиях AutoCAD). Затем следует перебор примитивов в наборе и определяется его тип и выполняются операции формирования списков координат и других данных для каждого из 5 *характерных* типов:

- разомкнутая POLYLINE со стрелкой в последнем сегменте — инцидентная дуга СП;
- TEXT, начинающийся на цифру — кратность инцидентной дуги;
- CIRCLE — вершина-позиция;
- замкнутая POLYLINE — вершина-переход;

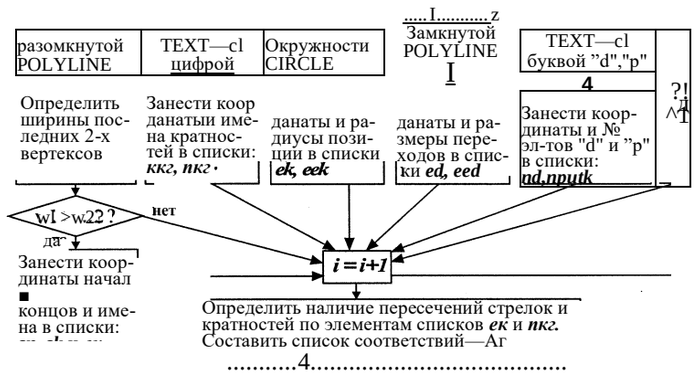
-TEXT с первой буквой “p” (“P”), “d” (“D”) или “g” (“G”) — надписи вершин СП.

Прочие типы игнорируются. Далее следует перебор отдельных послойных наборов и выполняются такие действия.

Определение кратностей инцидентных дуг. Каждой кратности присваивается условная пара характеристических отрезков, пересечение которых кривой POLYLINE некоторой инцидентной дуги (с учетом кривизны) фиксируется как кратность дуги большая 1 в списке соответствий кривых кратностям. Пересечение одной кривой двух кратностей и наоборот — ошибка.



В случае:



Определить № ближайшего перехода из **ed** к каждому тексту в **nd** перебором расстояний между ними - **vd.**

Найти координаты начал стрелок на расстоянии не более допустимого от переходов **vd,** и их имена

Найти координаты ближайших позиций к концам найденных стрелок и их № - **vp**

По спискам **vd, vp, кг** образовать структурированный список обратных инциденций уф вида **(£d, p)кг)**

Определить № ближайшей позиции из **ек** к каждому тексту в рр перебором расстояний между ними - **vp'.** Одновременно, по цвету, создать список ЖГСМ - **gen**

Найти координаты начал стрелок на расстоянии не более допустимого от позиций **vp'**, и их имена

Найти координаты ближайших переходов к концам найденных стрелок и их № - **vd'**

По спискам **vd,', vp', кг** образовать структурированный список обратных инциденций **vpd** вида ((р, **d)kr)**

Добавить списки У/Х/ и уф в общие списки **vpdo** и **vdpo**

Проверить наличие одинаковых № позиций и переходов.

Найти максимальные № позиций и переходов и распечатал данные из списков **vpdo, vdpo и gen** в файлы **dddd, pppp и gen**

Q Конец)

Рис.Б. 1. Блок-схема алгоритма программы получения матриц инцидентности

Определение номеров переходов и позиций. Перебор расстояний от переходов до надписей вершин на “d” (“D”) и выбор минимального. Формирование структурированного списка соответствий вида (№“d” координата “d”). Аналогично формируется структурированный список соответствий для позиций (независимо — “p” (“P”) или “g” (“G”)), номера ЖГСМ фиксируются в отдельный список.

Определение обратных отношений инцидентности. Определяются координаты начал дуг (стрелок) на расстоянии не более заданного от замкнутой POLYLINE перехода и их имена связываются в список с № перехода. Для каждой такой стрелки определяются номера позиций на расстоянии не более заданного и формируется структурированный список вида ((№ перехода, № позиции) кратность).

Определение прямых отношений инцидентности. Выполняется аналогично обратным и формируется структурированный список вида ((№ позиции, № перехода) кратность).

Далее формируются и после каждого послойного набора дополняются общие структурированные списки видов ((№ позиции, № перехода) кратность) — vpdo для прямых инцидентностей и ((№ перехода, № позиции) кратность) — vdpo для обратных инцидентностей.

По исчерпании всех послойных наборов следуют процедуры проверки корректности сформированных списков, нахождения максимальных номеров позиций и переходов для определения размерностей матриц инцидентности и распечатка последних в текстовые файлы **pppp** и **dddd,** а также распечатка номеров позиций ЖГСМ в файл **gen.**

Далее следует листинг программы.

•ПРОГРАММА ПОЛУЧЕНИЯ ИСХОДНЫХ ДАННЫХ МОДЕЛИРОВАНИЯ ПО ЧЕРТЕЖУ СЕТИ ПЕТРИ

```
(setq nn (ssget)
    Inn (sslength nn)
    gen nil
    rah 0
    linb nil
    vdpo nil
    vpdo nil
   tko nil
   odnp nil
   odnd nil
   про nil
   ndo nil
   prr nil
      i -1
    layp '(8 ." 1 ");заданный уровень с общими переходами подсетей
   layO layp ;поиск первого уровня 1ay0 не совпадающего с layp
   layO (while(equal layO layp)
```

```
            (setq i(l+ i)layO(assoc 8(entget(ssname nn i)))))
     sost (list layO)
    cmde (getvar ''cmdecho")
      jO
      o 0)
(setvar "cmdecho" 0)
;создание послойных наборов (слой " 1" + слой i)
(while (<= j rah)
(setq i 0
     nnb (ssadd)
     linb (cons nnb linb) ;список послойных наборов
     nb (read(strcat "nab"(itoa j))))
(set nb nnb)
  (While (< i Inn)
    (setq e (ssname nn i)
      layl (assoc 8(entget e))
        i(l+i))
    (cond((or(equal layl layO)(equal layl layp))(ssadd e nnb))
    ((and(null(member layl sost))(not(equal layl layp)))
    (setq sost (cons layl sost) rah (1+ rah)))
    (t nil))
  )
  (setq layO (nth j sost) j (1 +j))
)
(while (<= o rah)
(setq i 0
      p nil
     pt nil
     dk nil
      n (eval(read(strcat "nab"(itoa o))))
      I   (sslength n)
     nxx nil
     en nil
     ek nil
     ed nil
     eed nil
     eek nil
     kk nil
     kn nil
     kr nil
     kkr nil
     nkr nil
     nd nil
     nd2 nil
     np nil
     np2 nil
     vp nil
    vd nil
    vdp nil
    vpd nil
    sn nil
    sk nil
    tk nil
    tt nil
    xe nil
    ttt nil
    znil
    imo nil)
  обработка 2-мерных ''LWPOLYLINE":
(while (< i l)(setq nxx(cons (ssname n i)nxx)i(l+ i)))
(setq pr nxx)
(foreach xl pr(if(eq(cdr(assoc 0(entget xl)))"LWPOLYLINE")
            (progn(setq pr nil)
                 (if(null prr)(progn(setq prr t)
            (рготрГ'\пКонверсия полилиний: 2D-3D, прямых в кривые...")))
                 (ргтс"\пёыющ ")
(foreach x nxx
```

```
(setq xe (entget x) tt (cdr (assoc 0 xe)))
(if(equal tt ''LWPOLYLINE")
(if(= l(cdr(assoc 70 xe)))
    (progn(setq xx nil ly (assoc 8 xe))(mapcar '(lambda(x)
        (if(=(car x)10)(setq xx(cons(cdr x)xx)))) xe)
        (foreach xn linb(ssdel x xn))(command ''erase" (ssadd x) "")
        (command "_3dpoly" (car xx)(cadr xx)(caddr xx)(last xx)"c")
        (setq lsl(entget(entlast)))(entmod(subst ly(assoc 8 lsl)lsl))
        (foreach xn linb(ssadd(entlast)xn)))
    (command''pedit''(ssadd x)"f" "x"))))
(princ (1+ o))(prompt" - Ok!")))))

определение ширин последних вертексов
(setq i 0)
(while (< i 1)
(setq e (ssname п i)
      II   nil
      tt (cdr (assoc 0 (entget e))))

(if (equal tt "LWPOLYLINE")(prompt"\nERROR: No convert LWPOLYLINE !!"))

(if(equal tt ''POLYLINE") ;выделение имен крайних вертексов
(setq e (entget (nth (while (Null (equal(cdr(assoc 0(entget e)))"SEQEND"))
                        (setq e (entnext e)
                            11 (cons e 11)
                            j2)
                    )
             11)
      )
    en (cons (nth 211) en)                  последние вертексы
    kn (cons (entnext (ssname n i)) kn) ;первые вертексы
)
(setq en (cons e en)
      kn (cons e kn)
      e (entget e)
)
    )
      (setq p (cons (list (cdr (assoc 40 e)) (cdr (assoc 41 e))) p) .ширины
           z (cons (cdr (assoc 70 (entget (ssname n i)))) z) ;замыкания
          ttt (cons tt ttt)                          ;типы примитивов
           1   (1+0
           j0)
)
(setq i 0)
формирование списков координат стрелок, кругов, переходов, №, кратностей
(while (< i 1)
    (cond       ((and(eq      (nth      i      ttt)      "TEXT")      проверка      на      цифру      -      кратности
           (>(atoi(cdr(assoc 1 (entget(nth i en)))))0)
          )
      (setq ppt (entget (nth i en))
           pt (list(+(cadr(assoc 10 ppt))(*(strlen(cdr(assoc 1 ppt)))
                                    (cdr (assoc 41 ppt))
                                    (cdr (assoc 40 ppt))0.92)
                )
                (+(caddr (assoc 10 ppt))(cdr (assoc 40 ppt))) 0.0
             )
           pt (polar(cdr(assoc 10 ppt))
                 (+(cdr (assoc 50 ppt))(angle(cdr(assoc 10 ppt))pt))
                 (distance (cdr(assoc 10 ppt)) pt)
              )
;координаты отрезков текстов кратностей
           kkr (cons (list(cdr(assoc 10 ppt))pt) kkr)
           nkr (cons (nth i en) nkr)
      )
      )
      ((and (eq (nth i ttt) "POLYLINE")
            (> (car (nth i p)) 0)
```

```
          (= (cadr (nth i p)) 0.0) ;стрелки
 )
       (setq ek (cons (list
                 (cdr (assoc 10 (entget (nth i kn))))
                 (cdr (assoc 10 (entget (entnext (nth i en)))))
                    )ek)
             sn (cons (nth i kn) sn)
             sk (cons (nth i en) sk)
       )
)
 ((eq (nth i ttt) ''CIRCLE") ;круги
 (setq kk (cons (cdr (assoc 10 (entget (nth i en)))) kk)
     eek (cons (list (cdr(assoc 10 (entget (nth i en))))
                 (* 1.3(cdr(assoc 40 (entget (nth i en))))))eek)
     imo (cons (nth i en) imo)
 )
 )
 ((and(eq (nth i ttt) "POLYLINE")
     (or (= (car (nth i p)) (cadr (nth i p)) 0.0)
        (> (cadr (nth i p)) 0.0)
     )                              прямоугольники
     (or(= (nth i z) 1)(= (nth i z)9)) проверка на замыкание
 )
 (setq pl (cdr (assoc 10 (entget (entnext (ssname n (-1 i 1))))))
     p2 (cdr(assoc 10(entget(entnext(entnext(entnext
                                  (ssname n (-1 i 1))))))))
     ed (cons (polar pl (angle pl p2) (/ (distance pl p2) 2))
              ed)
     eed (cons (list (polar pl (angle pl p2)(/ (distance pl p2) 2))
              (* (/ (distance pl p2) 2) 1.2)) eed)
 )
)
((and(eq (nth i ttt) "TEXT") проверка на букву + цифру - вершины
    (Z=(atoi(substr (setq tx (cdr (assoc 1 (entget (nth i en))))) 2))0)
    (- (atoi(cdr (assoc 1 (entget (nth i en))))) 0)
 )
 (setq pt (polar (cdr(assoc 10 (entget (nth i en))))
                 (+(cdr (assoc 50(entget (nth i en))))0.785)
                 (*(cdr (assoc 40(entget (nth i en))))0.66))
       tk (cons pt tk)                          ;координаты
       nd (cons (if (or (eq (substr tx 1 1) "d")
                        (eq (substr tx 1 1) "D"))        ;№ переходов
                (atoi (substr tx 2))) nd)
       np (cons (if (or (eq (substr tx 1 1) "p")
                        (eq (substr tx 1 1) "P")
                        (eq (substr tx 1 1) "g")
                        (eq (substr tx 1 1) "G"))        ;№ позиций
                (atoi (substr tx 2))) np)
     gen (if(or(eq (substr tx 1 1) "g")
                (eq (substr tx 1 1) "G"))(cons(atoi(substr tx 2))gen)
                                         gen)
                                ;вспомаг. списки для про ndo
     nd2 (cons(if(and(not(equal(assoc 8(entget (nth i en)))layp))
                      (or (eq (substr tx 1 1) "d")
                          (eq (substr tx 1 1) "D")))
                 (atoi (substr tx 2))) nd2)
     np2 (cons(if(and(not(equal(assoc 8(entget (nth i en)))layp))
                      (or (eq (substr tx 1 1) "p")
                          (eq (substr tx 1 1) "P")
                          (eq (substr tx 1 1) "g")
                          (eq (substr tx 1 1) "G")))
                 (atoi (substr tx 2))) np2)

 )
)
(T nil)
)
```

```
(setq i (1+ i))
)

..... ..... определение кратностей дуг-..... —
(setq i 0)
    (while (< i (length sn))
        (setq k 0
            aa 1
            kr (cons (progn(while (< k (length kkr))
                    (setq e (nth i sn))
;поиск пересечений цифр и дуг с отслеживанием их кривизны —
                    (while (Null(eq e (nth i sk)))
                (setq phit (cdr(assoc 42(entget e)))
                    ptkl (cdr(assoc 10(entget e)))
                      e (entnext e)
                    ptk2 (cdr(assoc 10(entget e)))
                    ntkr (nth k kkr)
                      a nil
                      a (if(zerop phit)
                  (cond((inters ptkl ptk2(car ntkr)(cadr ntkr)))
                      ((inters (list(caar ntkr)(cadadr ntkr))
                                (list(caadr ntkr)(cadar ntkr))
                               ptkl ptk2))
                      ((eval 'a)))
                  (progn(setq kil (l+(abs(fix
                       (/(*(setq phi (*(atan(abs phit))4))
                          (setq rad (/(distance ptkl ptk2)
                                   (sqrt(+(expt(sin phi)2)
                                        (expt(- l(cosphi))2))))))
                       (distance(car ntkr)(cadr ntkr)))))))
                   (if(minusp phit)(setq ptkO ptkl ptkl ptk2 ptk2 ptkO))
                       (setq cen (polar ptkl
                                 (+(setq ango(angle ptkl ptk2))
                                       (/(- pi phi)2))
                                    rad)
                             ango (- ango (/(- pi phi)2))
                             phi (/ phi (if (zerop kil) 1 kil))
                              ptkl ptk2
                       anil)
                       (repeat kil
                       (setq ango (- ango phi)
                       ptk2 (polar cen ango rad)
                              a
                       (cond((inters (car ntkr)(cadr ntkr)ptkl ptk2))
                           ((inters (list(caar ntkr)(cadadr ntkr))
                                 (list(caadr ntkr)(cadar ntkr))
                                 ptkl ptk2))
                           ((eval 'a)))
                            ptkl ptk2)
                       )(eval 'a)))

                  (if (not(eq a nil))
                   (setq aa(atoi(cdr(assoc 1 (entget(nth k nkr)))))))
                       )
                   (setq k (1+ k))
                         )(if(null aa) 1 aa)) kr)
               i(l+i)
          )
     )
     (setq kr (reverse kr))

;==================установление соответствия позиций и переходов=========

(setq k 0)
(while (< k (length tk))
      (setq i 0
```

```
          d nil
        ddn nil
        ddk nil)

; ----------обратные инциденции .....

   (cond ((numberp (nth к nd))
определение расстояний от текста до переходов
       (while (< i (length ed))
            (setq d (cons (distance (nth к tk)(nth i ed)) d)
                I (1+0)
       ) ;выбор min координаты перехода
            (setq d (nth (1-(length (member (apply 'min d)d))) ed)
                 I0)
   Определение координат начал связанных стрелок
           (while (< i (length ek))
                (setq j 0
                  ddk nil)
            (cond((< (distance d (car(nth i ek)))(cadr(assoc d eed)))
       Определение коорд. ближайшего круга (позиции) каждой стрелки
                (while (< j (length kk))
                  (setq ddk (cons(distance(nth j kk)(cadr(nth i ek)))ddk)
                       j (1+j)))
              (setq ddk (nth(l-(length(member (apply 'min ddk) ddk)))kk)
                    j0
                   dk nil)
       ;список № перехода и этой позиции:
                  (while (< j (length tk))
                    (setq dk (cons (if (numberp (nth j np))
                                                (distance(nth j tk)ddk) 9999) dk)
                          j (1+j))
                   )
                (setq vp (nth (1-(length (member(apply 'min dk) dk))) np)
                    vd (nth k nd)
                    vdp (cons (list(list vd vp) (nth i kr)) vdp))
                        Описок обр. инциденций
                 )
                 (t nil)
           )
                 (setq i (1+ i))
          )
       )
■ ----------прямые инциденции ----------
        ((numberp(nth k np))
      Определение расстояний от текста до позиций
      (while (< i (length kk))
            (setq d (cons (distance (nth k tk)(nth i kk)) d)
                i(l+i))
      ) ;выбор ближайшей (min) координаты позиции
      (setq d (nth(setq gg(l-(length(member(apply 'min d)d))))kk)
      проверка цвета, формирование списка позиций ЖГСМ
        gen (if(and(equal(setq as62(assoc 62(entget(nth gg imo))))'(62 . 1))
                  (boundp 'as62)
                  (null(member(nth k np)gen)))
              (cons (nth k np) gen)gen)
         I0)
        определение координат начал связанных стрелок
          (while (< i (length ek))
                (setq j 0
                  ddk nil)
           (cond((< (distance d (car(nth i ek)))(cadr(assoc d eek)))
        Определение коорд. ближайшего перехода каждой стрелки
              (while (< j (length ed))
                (setq ddk (cons(distance(nth j ed)(cadr(nth i ek)))ddk)
                     j(l + j)))
            (setq ddk (nth(l-(length(member (apply 'min ddk) ddk)))ed)
                  j0
```

```
                dk nil)
          ;список № позиции и этого перехода:
              (while (< j (length tk))
                (setq dk (cons (if (numberp(nth j nd))
                                   (distance (nth j tk) ddk) 9999) dk)
                               j (1 + j)))
               (setq vd (nth (1-(length (member (apply 'min dk) dk))) nd)
                vp (nth k np)
                vpd (cons (list(list vp vd) (nth i kr)) vpd))
               )
               (t nil)
            )
              (setq i (I-ь i))
          )
        )
        (T nil)
    )
(setq k (1+ k))
)
(setq o (1+ o)
  vdpo (append vdp vdpo)
  vpdo (append vpd vpdo)
   nd (if(= o l)nd nd2)
   np (if(= o l)np np2)
   ndo (append nd ndo)
   про (append np про)
   tko (append tk tko))
)
проверка на одноименность
(setq 1 (length про)
      o 0)
(while (< o I)
  (setq nop (nth o про)
      nod (nth o ndo)
      nto (nth o tko)
      o(l+o))
  (if(and nop
        (or(member nop(cdr(member пор про)))
          (assoc nop odnp)))
     (setq odnp (cons(list nop nto)odnp)))
  (if(and nod
        (or(member nod(cdr(member nod ndo)))
           (assoc nod odnd)))
          (setq odnd (cons(list nod nto)odnd)))
)
(if(boundp ’prr)
  (prompt"\nnНРОВЕРЬТЕ И УТОЧНИТЕ РАСПОЛОЖЕНИЕ ЦИФР КРАТНОСТИ НА ДУГАХ!!!”))
  .....распечатка данных в файлы ..... .....
(setqj 1
   fd (open "dddd” "w")
   fp (open ”pppp” ”w")
   fg (open "gen" "w")
   ma (apply ’max (subst 0 nil ndo)))

(if odnd(prompt "\nВНИМАНИЕ!’.Обнаружены одноименные переходы!:"))
(while (<=j ma)
   (setq i 1)
   (princ "\n" fp)
  (while (<= i (apply 'max (subst 0 nil про)))
   (princ (if (null (assoc (list i j) vpdo))
        "0"
        (cadr(assoc (list i j) vpdo))) fp)
   (princ "\t" fp)
   (setq i (1+ i))
  )
  (if odnd(foreach x odnd(if(=(car x)j)(print x))))
   (setqj (1 + j))
```

```
(setq j 1
   mx (apply ‘max (subst 0 nil про)))
(if odnp(prompt ”\nВНИМАНИЕ!!Обнаружены одноименные позиции!:”))
(while (<= j mx)
   (setq i 1)
   (princ ”\n” fd)
(while (<= i (apply ’max (subst 0 nil ndo)))
   (princ (if (null (assoc (list i j) vdpo))
         ”0 ”
         (cadr(assoc (list i j) vdpo))) fd)
   (princ ”\t” fd)
   (setq i (1+ i))
)
   (if odnp(foreach x odnp(if(=(car x)j)(print x))))
   (setqj (1 + j))
)
(if gen
(progn
(setqj 1)
   (princ (nth 0 gen) fg)
(while (< j (length gen))
   (print (nth j gen) fg)
   (setqj (1 + j))
)))
(close fd)(close fp)(close fg)
(prompt "\nПозиций всего: ”)(princ mx)(princ ” ’’)
(prompt "Переходов всех: ")(princ ma)(princ " ’’)
(prompt "Позиций ’ЖГСЧ’: ")(princ (length gen))
(setvar "cmdecho" cmde)

            (prinl)
```

ПРИЛОЖЕНИЕ В

Программа испытания моделей узлов коммутации

Блок-схема алгоритма программы представлена на рис В.1. Для написания ее использован язык программирования AutoLISP, который встроен в графический редактор AutoCAD. Ниже следуют пояснения ней.

В начале программы следуют определения пользовательских функций, которые в дальнейшем применяются в программе: независимой генерации случайных чисел с различными законами распределения, транспонирования матричных списков, корректного считывания данных из текстового файла, обработки временных переходов.

Затем производится считывание данных из заранее подготовленных файлов: pppp и dddd — матрицы прямых и обратных инциденций, pri — вектор приоритетов срабатывания переходов, vr — вектор начальной разметки, tim — вектор временных задержек переходов, gen — № ЖГСМ и данные о законах распределения ЖГСЧ. Задаются — параметр потока первичных заявок X и допустимая вероятность $p_{доп}$ попадания более 2 вызовов за временной шаг моделирования At.

Далее следуют подготовительные процедуры для моделирования:

- вычисление временного шага моделирования (такта работы СП) At;
- пересчет временных задержек переходов в такты;
- создание начальных списков-очередей, которые состоят из нулей и имеют длины большие на 1, чем временные задержки переходов в тактах, а имена конструируются в форме *vi*, где *i* — номер перехода;
- формирование списка приоритетности обработки переходов на основе данных из файла pri, начальная часть списка — номера приоритетных переходов, остальная — номера бесприоритетных;
- формирование списков номеров выходных переходов ЖГСМ и упорядочение их по возрастанию — конструкция имен списков *sppj,j* — номер ЖГСМ.

Далее следует запрос числа тактов работы модели до распечатки результатов и начинается основной цикл программы по тактам.

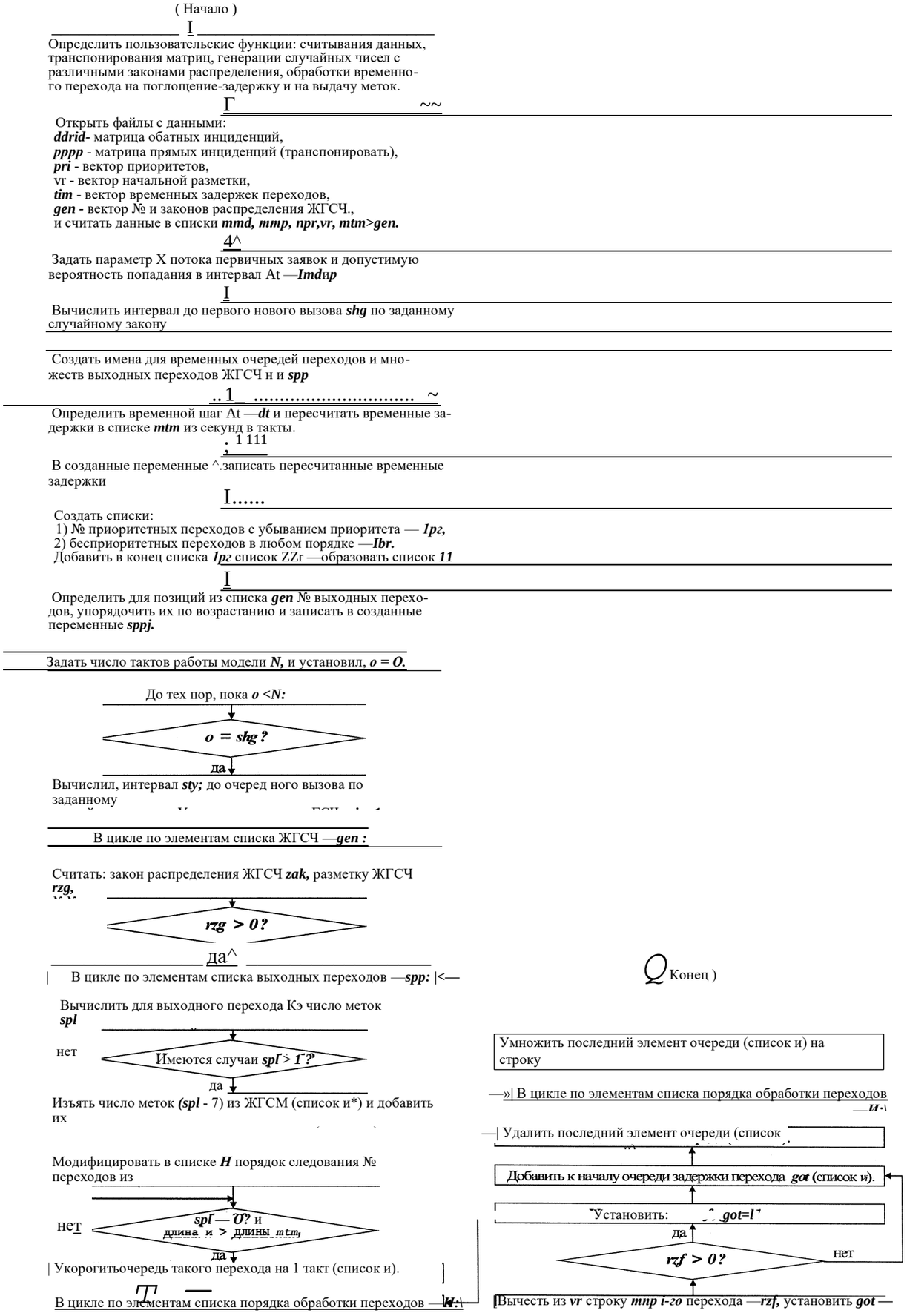


Рис.В. 1. Блок-схема алгоритма универсальной программы испытания моделей

В каждом цикле проверяется текущий номер такта на равенство ранее вычисленному моменту поступления новой заявки — в случае равенства в ГСМ добавляется 1 метка и вычисляется очередной новый момент поступления. Затем последовательно обрабатываются позиции ЖГСМ — при ненулевой ее разметке вычисляются номера срабатывающих выходных переходов по каждой метке, в соответствии с данными из файла gen, модифицируется список приоритетности обработки переходов для реализации правильного поглощения меток. В случае выпадения более одной метки на переход, очередь последнего принудительно удлиняется путем добавления метки в конец. В противном случае длина очереди сравнивается с нормой и при превышении ее — укорачивается на 1 метку.

После этого в циклах по элементам списка приоритетности обработки переходов (оператор foreach) выполняются функции обработки временных переходов, определенные в начале программы. Первая функция выполняет векторную операцию вычитания строки матрицы прямых инцидентностей, соответствующей текущему номеру перехода, из вектора разметки. Если результат не содержит отрицательных компонент (переход готов к срабатыванию), то вектор разметки устанавливается равным результату, и в “хвост” очереди текущего перехода добавляется “ 1 ” — изъятие меток. В противном случае вектор разметки остается неизменным и добавляется “0”. Затем удаляется последний элемент очереди, в результате чего все ее содержимое сдвигается на один такт. Вторая функция выполняет векторные операции умножения строки транспонированной матрицы обратных инцидентностей, соответствующей текущему номеру перехода, на последний элемент очереди (0 или 1) и вычитания результата из вектора разметки — выдача меток.

По окончании заданного числа тактов распечатывается вектор разметки, подсчитывается и распечатывается число вызовов, находящихся на обслуживании, и время моделирования.

Далее следует листинг программы.

```
;ф-ция транспонирования матричного списка (2-х мерн.)
(defun tsp (mmdd I Ins Inr md k i mdl)
(setq Ins (length mmdd)
      Inr (length (car mmdd))
       k(l-lnr))
(while(>= k 0)(setq mdl nil i (1- Ins))
(while(>= i 0)(setq mdl(cons(nth k(nth i mmdd))mdl))(1- i)))
```

```
(setq md (cons mdl md) к (1- k)))(eval 'md)
)

;ф-ция считывания данных из файлов с матрицами инциденций

(defun sch (ff /11 sst nst st)
(setq sst 111 nil)
(while sst
(setq nst nil)
(while (and(/= sst 10)(boundp 'sst))
(setq st "" sst (read-char ff))
   (while(and(/= sst 47)(> sst 45)(< sst 58))
   (setq st (strcat st(chr sst))
         sst (read-char ff))
   )
(if (/= st "")(setq nst (cons (read st)nst)))
)
(setq sst (if sst t)
        11 (if(boundp 'nst)(cons(reverse nst)ll)ll))
)
(close ff)
(reverse 11)
)


;ф-ция разовой генерации равномерно распределенных чисел
;от 0 до 1; ng - строка с № независимого генератора (натуральное)

(defun rnd (ng / c X r pc)
(setq r (read (strcat ''randomiz" ng))
       c (read (strcat ''dodatok" ng))
      pc (read (strcat "perekluc" ng)))
(if(null(eval r))(set r(atof(strcat "0"          ng)))); для переустановки X
(if(null(eval c))(set c 3147))
(if(null(eval pc))(set pc 0))
(setq X (+(fix (*27181 (eval r)))17)); переустановка X
(cond ((>= (eval c) 32091)(set pc 1))
        ((<= (eval c) 219) (set pc 0))) ;зададим пределы изменения 'c
   (set r (/(float(setq X (abs(rem(+(* 4029 X)(eval c))32767))))32767))
   (set c (if (zerop(eval pc))(l+(eval c))(l-(eval c))))
   (eval r)
)

;ф-ция разовой генерации целых чисел с экспоненциальным распре-
делением (аргументы: параметр Imd, dt - временной шаг,
;размерность dt д.б. согласована с Imd)

(defun rexc (Imd dt ng / u)
(setq u (rnd ng)
       u (l+(fix(/(if(zerop u)(/ 40(float lmd))(/(-(log u))lmd))dt))))
)

;ф-ция разовой генерации целых чисел со ступенчатым распреде-
лением (аргумент - список 1s вероятностей pO,pl,p2,...pN для
;первыхN целых полож. чисел 0,l,2,...,N)

(defun rst (Is ng / pr i le r)
(setq pr (rnd ng)
         I0
        le (length Is)
        sm 0.0000)
   (while (< i le)
   (setq r (if (and (>= pr sm)(< pr (+(nth i Is) sm)))i r)
        sm (+ (nth i Is) sm)
         I (1+i))
   )
   (if(<(abs(- sm 1.0)) 1 e-15)(if r r i)
          (progn(print"Uncorrect probability list!")(princ ls)(princ"?")))
```

```
)

;ф-ция обработки временного перехода по матрицам прямых инциденций
;mmp, разметки vr, и задержек tim (поглощение и задержка меток)

(defun perl (d / rzf got chas) ;d - № перехода
(setq rzf (mapcarvr (nth (1- d) mmp))
      got (apply 'and(mapcar 'not(mapcar 'minusp rzf)))
      chas (read (strcat "v" (itoa d))))
(if got (setq vr rzf))
(set chas(cons(if got 1 O)(reverse(cdr(reverse(eval chas))))))
)

;ф-ция обработки временного перехода по матрицам обратных инциден-
;ций mmd и разметки vr (выдача меток)

(defun per2 (d / chas)
(setq chas (last(eval(read (strcat "v" (itoa d))))))
        vr (mapcar'+ vr
                 (mapcar'(lambda(x)(* x chas))(nth(l- d)mmd))))
)

•========= ОСНОВНАЯ ПРОГРАММА ==============

очистка, считывание и подготовка исходных данных

(setq fd (open "dddd" "r")
      fp (open "pppp" "r")
      fpr (open "pri" "r")
      fr (open "vr" "r")
      ft (open "tim" "r")
      fg (open "gen_" "r")
       mmd (tsp (sch fd))
       mmp (sch fp)
       mpt (tsp mmp)
       mpr (sch fpr)
       mtm (sch ft)
      vr (car(sch fr))
      gn (sch fg)
      gen (car(tsp gn))
      Ivr (length vr)
      Ing (length mmp)
      Igen (length gen)
        jO
        I0
        oO
       Imd 2.51
        pO.l)
(while (< i (max Ivr Ing))
   (setq v(read(strcat "v"(itoa (1+ i))))
      pp(read(strcat "spp"(itoa (1+ i))))
       i(l+ i))
;(print i)(princ(eval v))
(set v nil)
(set pp nil)
)(setq i 0)
определение временого шага dt при интенсивности Imd и
допустимой вероятности p, числа шагов shg до сл. метки,
;персчет временных задержек из секунд в такты (mtm)

(setq dt (/(-(log(- 1 p)))lmd)
      shg (rexc Imd dt "1")
      mtm (mapcar '(lambda(x)(subst(fix(+(/(cadr x)dt)0.5))(cadr x)x)) mtm))

установка последовательности обработки переходов в порядке приоритетности по
;матрице приоритетов mpr (список И), начальных временных очередей переходов.
```

```
(setq len (length mpr) Ipr nil)
(while (< j Ing) ;цикл по переходам
(setq i 0
      ve nil
      ez nil
      mx nil
    pds nil
    mprO mpr
    chas (read (strcat ”v” (itoa(l + j))))
    cha (eval chas)
      q (cadr(assoc(l + j)mtm)))
(while (> q 0) (setq cha (cons 0 cha) q (1- q)));co3flanne 0-очередей
(set chas cha)
определение № перехода с максимальным приоритетом mx
  (while (< i len)
  (setq ve (cadr(setq pdd(nth i mprO))))
   (if (and(> ve ez)(null(member (car pdd)lpr)))(setq mx (car(setq pds pdd))
                                          ez ve))
  (setq i (1+ i))
  )
   (setq Ipr (if(< j len)(cons mx lpr)lpr) ;список №№ приоритетных переходов
       mprO (subst (list mx 0) pds mprO) ;замена максимального pr нулем
        j (1+j))
)
(setqj 1 y=l при нумерации переходов с ”1”!!!
      Ibp nil)
(while (<= j Ing) формирование списка №№ бесприоритетных переходов
  (if (null (member j Ipr)) (setq Ibp (cons j Ibp)))
  (setqj (1 + j))
)
(setq li (reverse(append Ibp Ipr))
      pO)

формирование spp№ - списков переходов прямо инцидентных "gen”
(foreach x gen
          (setq spp nil spn nil ip (nth(l- x)mpt))
          (while (< p Ing)
              (if(zerop(nth p ip))nil(setq spp(cons(l+ p)spp)))
              (setq p (1+ p)))
;упорядочим spp по возрастанию №№:
          (setq p 0 pp(read(strcat ”spp”(itoa x)))lll(length spp))
          (while(< p 111)
           (setq mx (apply ’max spp)
              spn (cons mx spn)
              spp (subst 0 mx spp)
                p (1+p))
          )
          (set pp (list spn spn)))
(setq number (getint ’’Enter 1/4 number of cycles: ”)
      number (+ number number number number))
(setq tttt (rtos(getvar ”cdate’’)2 6) az 0)

;общий цикл

(print vr)
(while (< o number)
(setq k 0
      j0)

обработка позиции ГСЧ (pl): генерация 1 метки через случ. число шагов

(if (= o shg)
    (setq shg (+ shg(rexc Imd dt ”1”)) vr(cons (l+(car vr))(cdr vr))))

обработка позиций ЖГСЧ: рапределение меток по выходам по случайному
;закону, создание и ликвид. очередей для предположения о неодновремен-
ное™ заявок, коррекция разметки ”gen” для использ. станд. ф-ции perl
```

```
установка з-на распределения локальных № выходов из файла gen
;цикл по позициям ”gen”:
(foreach z gen
  (setq zak ’(rst (cdr(assoc z gn))(itoa z))
      rzg (nth (1- z) vr)
      sppi (read(strcat ”spp”(itoa z)))
      spp (car(eval sppi))
      sppl (last(eval sppi))
      Ipp (length spp)
      nsp sppi
        m 0
      dob 0)
(if(> rzg 0)
   (progn(setq spl (mapcar ’(lambda(x)(cons x 0))spp))
/выброс" сл.№ перех.,коррекция очередей: цикл по внутренней разметке "gen"
   (while (< m rzg)
    (setq nn (nth (eval zak)sppl);cny4. число с пересчетом в факт. № перехода
       nsp (mapcar’(lambda(x)(if(/= nn x)x))пзр);выделим выбр. переход
       och (read (strcat "v" (itoa nn)))
       ppi (assoc nn spl)
        m (1+ m)                добавим 1 к выпавш. перех
       spl (subst (setq ppi 1 (cons nn(l+(cdr ppi)))) ppi spl))
   (if(>(cdr ppi 1)0)
      (progn
      (setq dob (1+ dob)) ;размер коррекции разметки "gen"
      (set och (cons l(eval och))))) ;удл. очереди при одновременных метках
   )
определение порядка обработки переходов "gen" при ненулевой разметке
   (setq i 0 rsp (mapcar ’(lambda(x)(cons(cdr x)(car x)))spl) npr nil sp2 nil)
   (while(< i Ipp)
     (setq a (assoc(apply ’max(mapcar ’car rsp))rsp)
        npr (cons(cdr a)npr)
        rsp (subst (cons(~(car a ))0) a rsp)
          i(l+i)))
   (setq npr (reverse npr) spro (mapcar ’(lambda(x y)(cons x y)) spp npr))
   (set sppi (list npr sppi))
;модификация последовательности прироритетной обработки переходов li
   (setq a nil
      li (mapcar ’(lambda(y)(foreach x spro
                      (if(= y(car x))(setq a(cdr x))))
                                          (setq a(if(or(null a)(= a b))y a)b a))li))))
;укорочение очередей переходов "gen" при отсутствии в их хвосте меток
  (foreach x nsp
   (if(and(numberp x)(setq vv(eval(setq v(read(strcat "v"(itoa x))))))
        (>(setq lvv(length vv))
          (1 +(if(setq xx(last(assoc x mtm)))xx(if(zerop(car vv))0 Ivv)))))
      (set v (cdr vv))))
  ;коррекция разметки "gen": вычет dob из vr
   (setq m 0
      vr (mapcar’(lambda(x)(setq m(l+ m))(if(= m z)(- x dob)x))vr))
)

;цикл поглощения меток ------------------------------
(foreach n li (perl n))

;цикл выдачи меток ------------------- ------- --------
(foreach n li (per2 n))

;(if(> (nth 62 vr)(+(nth 14 vr)(nth 65 vr)))
; (print (list v30 v31(nth 62 vr)(nth 65 vr)"pl5="(nth 14 vr)"o="o)))
   (setq o (1+ o))
)

(setq m 0 mm 0 ssss 0 sss 0)
;подсчет меток в первых (m-1) позициях (подсеть алгоритма)
(while(< m 14)(setq m(l+ m)ssss(+(nth m vr)ssss)))
```

```
;подсчет меток в первых (mm-i) переходах (подсеть алгоритма)
(while(< mm 33)(setq mm(l+ mm)
        sss(+(apply'+(cdr(reverse(eval(read(strcat"v"(itoa mm)))))))sss)))
(print vr)
(princ"\n=p юсёысцштрэшш: ")(princ(+ ssss sss))
;(princ"; THxE.g23: ")(load"v")

(prїпc"\пTExь ЕрсюЄу: ІяЕюуЕ./ Іьюф., c....")
(setq ttx(rtos(getvar "cdate")2 6))
(princ(setq tm
        (-(+(*(atoi(substr ttx 7 2))86400)
        (*(atoi(substr ttx 10 2))3600)
        (*(atoi(substr ttx 12 2))60)
        (atoi(substr ttx 14)))
       (+(*(atoi(substr tttt 7 2))86400)
        (*(atoi(substr tttt 10 2))3600)
        (*(atoi(substr tttt 12 2))60)
        (atoi(substr tttt 14))))))
(princ'7")(princ(* dt(l - o)))(princ"=")(princ(/ tm (* dt o)))
(princ)
```

ПРИЛОЖЕНИЕ Д

Использование сетевых моделей в инженерных целях

Д.1. Модель узла коммутации типа DX-200 емкостью 2048 №

Рассмотрим задачу оценки избыточности ресурсов подсистемы управления реального УК типа DX-200 емкостью 2048 номеров и определения минимального состава ресурсов при заданной пропускной способности Y=0.16 Эрл/АЛ. В качестве контролируемых параметров выберем:

1) вероятности превышения заданного времени ожидания сигнала “ОС” — $p_o(1)$ и $p_o(3)$ — соответственно 1 и 3 с,
2) вероятность превышения заданного времени ожидания при коммутации, то есть сигнала “КПВ” или “СЗ” — $p_к(2)$ — 2с [42,97].

Для упрощения данного примера будем считать, что УК обслуживает только местные соединения, телефонная периферия состоит из телефонных аппаратов с тональным набором номера, выполнение ДВО “переадресация” всегда занимает одно и то же время и одинаковое количество ресурсов, ресурсы датчиков голосовых сообщений “абонент не доступен” (при ДВО “Не беспокоить”) и “неправильный номер” не ограничены, прочие виды ДВО не востребуются, нумерация 4-значная, время ожидания абонентами сигнала ОС усреднено (3с). Рассмотрим вариант комплектации УК оборудованием DX-220 [12,56,77].

При описании ресурса абонентского модуля SUB необходимо предусмотреть 32 независимых источников нагрузки, каждый из которых создается 64-линейной группой абонентов с заданной интенсивностью удельной нагрузки (в примере Y=0.16 Эрл/АЛ). Для описания указанного ресурса можно использовать один из приемов, описанных в п.2.3, например многослойную модель абонентского модуля с числом слоев 2048:64=32. Однако с целью простоты и максимальной наглядности примера используем метод замены группы источников заявок эквивалентным источником, поток заявок которого идентичен суммарному потоку групп.

Ресурсными компонентами модуля является микроЭВМ семейства DMS ,
(“процессор абонентской сигнализации” — SSP), которая выполнена на базе стандартного 8-разрядного микропроцессора 8085, а также число доступных каналов СЛ ИКМ—

для каждой из 64-линейных групп — 30 каналов, всего 8х30=240 каналов. Последний имеет ограниченную доступность равную 30. Выходной поток заявок каждого 30-линейного пучка СЛ ИКМ является сглаженным [33], поскольку осуществляется сжатие 64:30. Суммарный поток всех СЛ ИКМ модуля SUB будет так же сглаженным, однако, для представления ресурса СЛ ИКМ как общего для всех АЛ, необходимо уменьшить число каналов в полно доступном пучке на столько, чтобы вероятность ожидания осталась неизменной. Для определения требуемой величины ресурса в модели можно воспользоваться методами теории телетрафика. Вероятность ожидания связана с интенсивностью нагрузки и числом линий в пучке общеизвестным вторым распределением Эрланга [33]. Во избежание громоздких вычислений по соответствующей формуле, на рис.Д. 1.1 приведена зависимость требуемого числа каналов полнодоступного пучка от интенсивности поступающей нагрузки для обслуживания простейшего потока заявок с вероятностью ожидания p=0,02% при показательном времени обслуживания и дисциплине с ожиданием, которая построена по соответствующей таблице (с экстраполяцией) [33]. По графику, например, можно определить число линий пучка v, которое необходимо для обслуживания нагрузки пучка из 512 линий Y=512х0,16=81,6 (Эрл) с таким же качеством (p=0,02%), как и при обслуживании нагрузки одного модуля Y=10,2 Эрл тридцатью линиями — на рис.Д. 1.1 — 8. В нашем случае общий пучок из 2048 линий создает нагрузку Y = 326,4 Эрл, которому при той же вероятности ожидания соответствует число обслуживающих линий полнодоступного пучка v=547,7. Следовательно в модели можно заменить 32 30-канальных пучка одним 548-канальным. Аналогично можно поступить и с ресурсом SSP, рассматривая в качестве обслуживающих линий пучка процессорное время в условных единицах (п.4.2.1), необходимое для обслуживания одного вызова. Различие в характере ресурсов в том, что каналы захватываются на все время обслуживания вызова, а процессорное время лишь на этапах установления и разрушения соединения. Данный факт отражается различиями в отношениях инцидентности соответствующих ресурсных позиций.

Другими ресурсными компонентами моделируемой системы, при введенных упрощениях, являются микроЭВМ типа DMS подсистемы управления совместно с приемниками и датчиками сигналов (см. п.2.3):

- SSU —УУ блока АИ,

- M — маркер,
- RU — регистр,
- CM — блок системных данных,
- PRBU — блок приемников тонального набора,
- TG(“ОС”), TG(“С3”), TG(“КПВ”) — цифровые тональные генераторы соответствующих сигналов.

SDL-диаграмма процесса обслуживания вызовов представлена на рис.Д. 1.2.

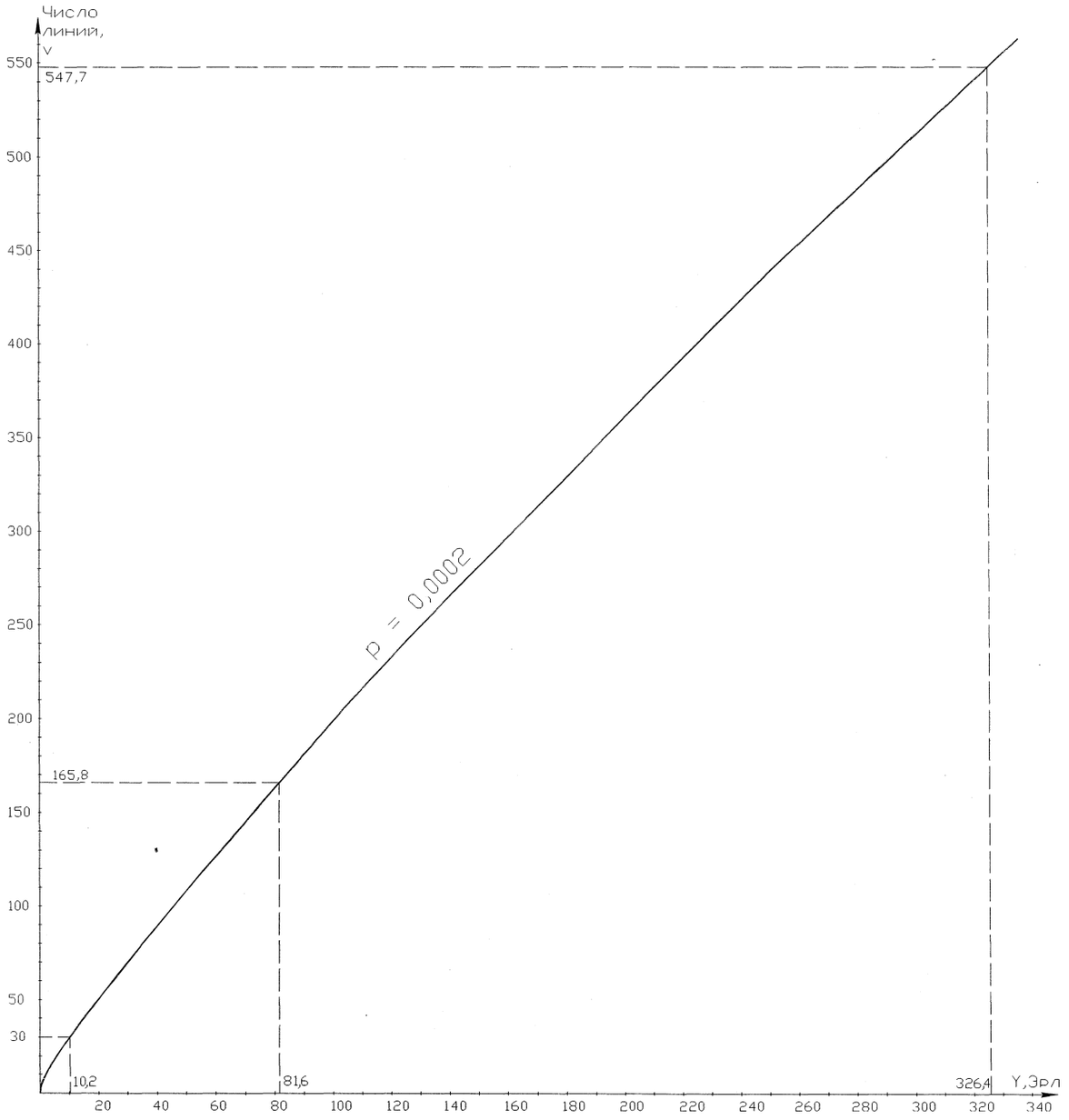


Рис.Д. 1.1. Пересчет числа каналов ИКМ СЛ в эквивалентный полнодоступный ресурс

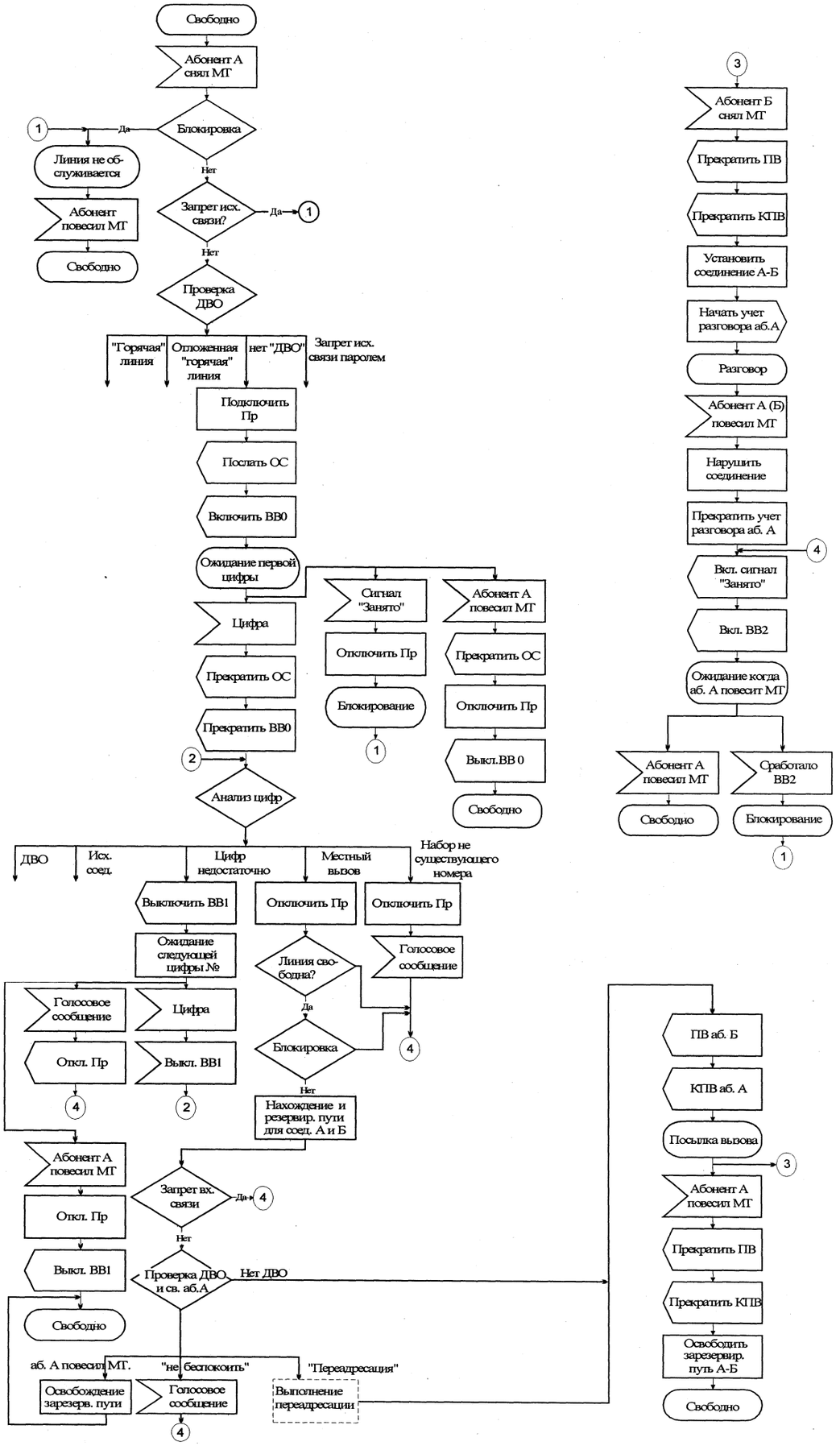


Рис.Д. 1.2. SDL-диаграмма местного соединения

Для оценки выбранных контролируемых параметров $\mathbf{p_0}$ и $\mathbf{p_к}$ необходимо в процессе функционирования модели зафиксировать общее число поступивших вызовов N_n и число вызовов, при обслуживании которых было допущено ожидание свыше заданного времени на этапах занятия N_o и установления соединения по набранному номеру N_K при дисциплине обслуживания с ограниченным ожиданием. Искомые вероятности определятся, как отношения:

$$Po = \frac{N_n}{N} Pк = — \sim \blacksquare \frac{N_n}{N}$$

В соответствии с методикой (этапы I—IV) создаем чертеж модели, в котором строим подсети алгоритма, внешнего окружения, ресурсов и анализа (приложение 4). Позиция pl моделирует внешний источник первичных заявок, позиции p24-p14 — этапы обслуживания; p64^-p73 —ресурсные компоненты УК, соответственно: SSP, каналы ИКМ-линий от SUB, SSU, PRBU, CM, TG(“OC”), TO(“КПВ”), TG(“C3”), M, RU; p75-p85 — позиции анализа: p85 и p86 подсчитывают число вызовов ожидание которыми сигнала ”OC” превысило соответственно 1с и 3с, а p87 — то же для сигнала КПВ и времени 2с, p84 — определяет общее количество вызовов, принятых на обслуживание, p82 — контроль числа вызовов, покинувших сеть, остальные позиции анализа (p75 — p83) — вспомогательные; g20, g23, g24, g26, g27, g32, g35-j-g38, g40, g49, g52, g60, g61 — ЖГСМ подсети внешнего окружения; остальные позиции моделируют условия сформированное™ тех или иных вторичных заявок или сигналов. Теперь (этап V), на основании статистических сведений 6 внешнем окружении, назначим временные задержки выходов ЖГСМ в файле ”Tim”. По техническим данным оборудования дополним этот файл временными задержками переходов подсети алгоритма, установим начальную разметку ресурсных позиций в файле “Vr”, соответствующую использованию одной микроЭВМ каждого типа, и добавим кратности дуг на чертеже (слой подсети ресурсов). В файле “Pri” назначим приоритеты так, чтобы адекватно отразить дисциплину с ограниченным ожиданием на этапе приема вызова: (pr(di)>pr(d32), pr(d92)>pr(d32)); затем логику приема цифр номера: (pr(di3)<pr(di4), pr(di3)<pr(di2), pr(di3)<pr(dn)); затем правильность подсчета N_o и N_K: (pr(d92)>pr(d9o)>pr(ds9), pr(d96)>pr(d9s)), приоритетность исполнения частных алгоритмов операционной системой (см. распечатку файла “Pri”). Далее (этап VI), сканируя чертеж, получим матрицы инциденций и закончим описание вероятностных распределений

ЖГСЧ указанием вероятностей их выходов в файле “Gen” по данным спектра занятия (табл. Д.1).

Таблица Д.1

Описание	**№ ЖГСМ**	1) **Вероятностный ряд** 2) **Временной ряд**
Время до I цифры № (3), отбой до НН(1), ВВ до НН(1)	24	0,62 0.3 0.05 0.025 0.005 2 7 0.8 1.5 15
Время до II цифры № (2), отбой (2), ВВ (1), несущ.№(1), ДВО(1)	26	0,61 0.3 0.035 0.01 0.005 0.04 0 0.5 2 0.7 2.5 15 0 0
Время до III цифры № (2), отбой (2), ВВ(1)	27	0,65 0.3 0.035 0.01 0.005 0.5 2 0.7 2.5 15
Время до IV цифры № (2), отбой (2), ВВ(1)	32	0,65 0.3 0.035 0.01 0.005 0.5 2 0.7 2.5 15
аб.Б свободен / занят	35	0.84 0.16 0 0
Соединительный путь есть / нет	36	0.98 0.02 0 0
Время прослушивания КПВ(4)	37	0.2 0.4 0.3 0.1 3 6 9 12
Ответ аб.Б / отбой аб.А	38	0,75 0.25 0 0
Время разговора (4 градации)	40	0,2 0.3 0.3 0.2 35 95 185 495
Время прослушивания С3(4), ВВ(1)	49	0,4 0.3 0.197 0.1 0.003 2 5 10 25 120

Для организации испытаний модели необходимо задаться значением интенсивности потока поступающих вызовов X (в $с^{1}$) и допустимой вероятностью $р_{доп}$ попадания более 2 вызовов за временной шаг моделирования At. В случае такого попадания вызов (вызова) “переносится” в первый свободный интервал (интервалы) At, следующий за данным. Экспериментально установлено, что при $р_{ДОГ1}$<0,1 такое преобразование практически не влияет на математическое ожидание и дисперсию полученных результатов.

Интенсивность X в данной постановке задачи должна быть определена по данным статистики и установленной величине удельной телефонной нагрузки на линию. Для этого определим среднее время занятия $t_{ср}$ абонентской линии, как сумму величин:

- среднее время прослушивания сигна ла ОС $t_{ос}$ (g24),
- среднее время набора номера $t_{НН}$ (g26,27,32),
- среднее время прослушивания сигнала КПВ (g37),
- среднее время разговора $t_{р}$ (g40),
- среднее время прослушивания сигнала СЗ (блокировки) $t_{СЗ}$ (g49),

с учетом вероятности попадания вызова на каждый этап (g35, g36, g38). Рассматриваемый в примере вероятностно-временной спектр занятий представлен в табл. Д.1. В скобках указано число выходных переходов ЖГСМ, модели (позиции, не существенные и малозначимые для определения $t_{ср}$, исключены, соответствующие им данные спектра имеются в файлах “Gen” и “Tim” — приведены ниже).

По табл.Д. 1, используя очевидные соотношения, можно рассчитать $t_{ср}$:

a) определим $t_{ос}$: $t_{ос}$=0,62-2+0,3-7+0,8-0,05+0,025-1,5+0,005-15=3,495(с)

b) определим Інн

I-II цифра: $t'_{НН}$=0,61 -0,5+0,3-0,2+0,035-0,7+0,01 -2,5+0,005-15+0,04-1 = 1,07(с),

II-ПІ цифра: $t''_{НН}$=0,65-0,5+0,3-0,2+0,035-0,7+0,01-2,5+0,005-15+=1,03(с),

III-IV цифра: f "нн=0,65-0,5+0,3-0,2+0,035-0,7+0,01 -2,5+0,005-15+=1,03(с);

с учетом вероятностей отбоя на этапе НН:

$t_{НН}$= Гнн+0,91-і"нн+0,91-0,95-Ґ”нн

$t_{НН}$= 1,07+0,91 • 1,03+0,91 -0,95-1,03=2,94(с)

c) определим їкпв’ Ікпв-0,2-3+0,4-6+0,3-9+0,1-12=6,9(с)

d) определим $t_{р}$: $t_{Р}$=0,2-35+0,3-95+0,3-185+0,2-495=190(с)

e) определим $t_{СЗ}$: ^=0,4-2+0,3-5+0,197-10+0,1-25+0,003-120=7,14(с)

определим $t_{ср}$ учетом вероятностей попадания вызова на соответствующий этап:

$t_{с}$ = toc + 0,97-t,,H + 0,97-0,84-,0,98-$t_{КпВ}$ + 0,97-0,84-,0,98-0,75-$t_{р}$ + ((1-0,97) + (1--0,97-0,84-,0,98) + (1-0,97-0,84-,0,98-0,75))-$t_{СЗ}$

$t_{ср}$ = 3,495 + 0,97- 2,94 + 0,97-0,84-,0,98-6,9 + 0,97-0,84-,0,98-0,75- 190 + ((1--0,97) + (1-0,97-0,84-,0,98) + (1- 0,97-0,84-,0,98-0,75))-7,14 = 130,16(с)

Среднее число вызовов m на АЛ в течении ЧНН очевидно определится как:

$$m = — 3600,$$
$$^{\wedge}cp$$

где Y — удельная интенсивность поступающей нагрузки в Эрл,

$t_{ср}$— выражено в с.

Тогда интенсивность поступающих вызовов определится выражением:

$$A = N\frac{m\ NY}{3600\ t_{c,,}} = \frac{}{}$$

где % — интенсивность поступающих вызовов в с,

N — емкость УК.

Для заданной пропускной способности УК:

$$Л = \frac{2048^{\wedge}16}{130,16} \quad 51\ (с.,)$$

Назначим начальную разметку СП по реальной конфигурации УК (см. табл. Д.2).

Таблица Д.2

SSP, *p64*	к. ИКМ от SUB, *p65*	SSU, *pбб*	PRBU, *p67*	CM, *p68*	TG(“OC”) *p69*	TG(“KIIB”), *p70*	TG(“C3”), *p71*	M, *p72*	RU, *p73*
18	548	128	64	16	32	32	32	20	64

Проводим моделирование. Разметки вычисляющих позиций р754-р83, полученные в результате испытаний модели в течении 10 ЧНН и определенные по ним величины вероятностей превышения соответствующих времен ожидания (при временном шаге моделирования At=0,041с) представлены в табл.Д.З.

Т аб лица Д. 3

р85	р86	р84	Pod), Ю-з	Po(3), Ю-з	р87	р84	Р«(2), Ю’
4	2	8835	0,45	0,22	0	8835	0
2	0	8955	0,22	0	0	8955	0
2	1	8838	0,23	0,11	0	8838	0
3	0	8754	0,34	0	0	8754	0
5	2	8886	0,56	0,23	0	8886	0
1	0	8734	0,11	0,11	0	8734	0
6	3	8964	0,67	0,33	1	8964	0,11
2	0	8903	0,22	0	0	8903	0
5	2	8868	0,56	0,23	0	8868	0
4	1	8843	0,45	0,11	0	8843	0

Как видно из таблицы, вероятности ожидания намного ниже норм $р^{н}_{о}(1)=0,005$; $р^{н}о(3)=0,001$; $р^{н}к(2)=0,005$, что вполне объяснимо, так как максимальная емкость одного блока АИ по данным разработчиков 3904№, то есть значительно больше принятой в примере. Отсюда вывод — УУ УК имеет значительную избыточность в ресурсах.

Для оценки степени уменьшения количества ресурсов необходимо контролировать текущую разметку ресурсных позиций. К сожалению средствами предлагаемого расширения СП не удается фиксировать экстремальные значения текущей разметки позиций, поэтому необходимо производить коррекцию начальной разметки с последующим уточнением контролируемых показателей УК в несколько приемов. В качестве начального ориентира целесообразно принять среднее значение величины ресурса, поглощенного к моменту снятия показаний в установившемся режиме $М_{П}ср=Мо-ХМi/1О$, исключая р65 (ИКМ-30/32). Табл. Д.4 иллюстрирует сказанное. Учтем запасом в 30-60% Мпср, что моменты превышения допустимых величин и моменты фиксации Mj случайны (по опыту моделирования).

Таблица Д.4

	р64	р65	рбб	р67	р68	р69	р70	р71	р72	р73
Мо	18	548	128	64	16	32	32	32	20	64
Mi	12	113	99	39	14	17	23	19	15	46
$м_2$	10	81	114	60	10	19	20	22	18	51
Мз	15	77	119	37	13	20	25	25	16	50
$М_4$	10	101	97	50	9	16	19	16	14	44
$М_5$	11	57	93	61	12	22	20	18	17	41
$М_6$	10	111	86	40	10	17	19	17	14	45
$м_7$	13	182	109	43	12	23	24	21	16	49
$м_8$	15	68	89	54	14	24	22	24	14	55
$м_9$	12	121	78	45	14	17	18	18	19	42
Мю	14	93	96	41	15	22	21	22	16	49
Мпср	5	548	30	17	4	13	11	12	6	17

По результатам проведенных испытаний назначим начальную разметку ресурсов равную (1,3-1,6)$М_{ПСР}$ по каждому ресурсу (с округлением - табл. Д.5), исключая р65 и повторим испытания.

Таблица Д.5

SSP, *рб4*	к. ИКМ от SUB, *р65*	SSU, *рбб*	PRBU, *р67*	CM, *р68*	TG(“OC”) *рб9*	TO(“КПВ”), *р 70*	TG(“С3”), *р71*	M, *р72*	RU, *р73*
18	548	128	64	16	32	32	32	20	64

Результаты определения вероятностей ожидания приведены в табл.Д.6.

ТаблицаД. 6

р85	р86	р84	**Ро(1), 10-3**	**Ро(3),** Ю-з	р87	р84	**Р-Р), 10’³**
48	25	8755	5,45	2,86	5	8755	0,57
89	31	8991	10,29	3,41	3	8991	0,33
92	21	8938	5,82	2,34	5	8938	0,56
104	31	8901	11,68	3,48	6	8901	0,67
44	22	8810	4,99	2,50	4	8810	0,45
75	29	8802	8,52	3,29	7	8802	0,80
53	19	8933	5,93	2,13	2	8933	0,22
47	18	8799	5,34	2,05	4	8799	0,45
121	29	8831	13,69	3,28	8	8831	0,90
54	21	8737	6,18	2,40	4	8737	0,46

Из таблицы видно, что **p_0(1)** и **p_0(3)** превышают допустимые значения $р^{н}_{0}$(1)=0,005 и $р^{н}$о(3)=0,001. Анализ разметок ресурсных позиций показывает, что наиболее “узкими” местами являются разметки р66(SSU) и р68(CM) (табл.Д.7).

Таблица Д.7

р64	р65	рбб	р67	р68	р69	р70	р71	р72	р73
7	74	0	11	0	7	7	8	7	9
7	141	9	9	2	7	9	12	9	15
8	95	4	15	1	12	7	11	8	6
6	64	1	6	0	9	11	8	2	14
8	188	8	9	2	6	12	11	4	18
7	100	5	17	4	11	7	13	8	10
5	137	17	14	3	7	8	7	8	11
3	78	0	8	0	4	3	5	3	10
6	68	1	10	0	9	6	4	2	4
5	89	0	9	2	9	4	12	5	12

Ясно, что для улучшения показателей функционирования необходимо увеличение начальной разметки этих позиций. Требуемую величину коррекции определяем подбо-

ром, для чего повысим начальную разметку Mo(p66) с 48 до 51 и Mo(p68) с 6 до 8 и повторим испытания. Результаты представлены в табл.Д.8.

Таблица Д.8

p85	p86	p84	**Ро(1),** Ю-з	**Ро(3), Ю-з**	p87	p84	**Рк(2),Ю-з**
21	8	8937	2,35	0,90	5	8937	0,56
32	9	8859	3,61	1,01	0	8859	0
11	2	8855	1,24	0,22	0	8855	0
19	1	9014	2,11	0,11	4	9014	0,44
16	5	8819	1,81	0,57	1	8819	0,11
27	7	8794	3,07	0,80	0	8794	0
47	9	8889	5,29	1,01	1	8889	0,11
25	5	8813	2,84	0,57	1	8813	0,11
18	4	8735	2,06	0,46	2	8735	0,23
38	7	8990	4,23	0,78	3	8990	0,33
	Рср		2,86	0,64			0,19

Как видно из таблицы, показатели УК практически в норме, поэтому можно считать задачу решенной после выполнения обратного пересчета разметки позиций в ресурсные характеристики. При необходимости более точного определения минимального состава ресурсов процесс можно повторить при большем числе испытаний. Сопоставление ресурсов до (р') и после (р") снижения разметки представлено в табл. Д.9 (8р,% — относительное уменьшение ресурса).

Таблица Д.9

	SSP, *p64*	к. ИКМ от SUB,p65	SSU, *pбб*	PRBU, *pб7*	CM, *p68*	TG("ОС") *p69*	ТО("КПВ"),p70	TG("СЗ"), p77	M, *p72*	RU, *p73*
Р'	18	548	128	64	16	32	32	32	20	64
Р"	8	548	44	28	6	19	16	18	9	28
8р,%	56	-	66	56	63	41	50	44	55	56

Таким образом оценка избыточности ресурсов составила 41 — 66 %. Результаты моделирования показывают, что подсистема управления моделируемого УК имеет значительную избыточность, особенно по блокам SSU, CM, RU, SSP. Это вполне объяснимо, поскольку ресурс SSU рассчитан на почти вдвое большую нагрузку (4096 АЛ при У=0,15Эрл/АЛ), блоки SSU реализован на стандартной микроЭВМ, ресурс которой невозможно уменьшить. Изучение практических ситуаций, возникающих на УК, подтверждает, что исключение двух из четырех блоков RU практически не ощущается на статистике

потерь. По результатам можно сделать практический вывод о том, что ресурс подсистемы управления по всем позициям, кроме TG(“ОС”) и TG(“С3”) достаточен для обслуживания почти вдвое болыпей'нагрузки.

Ниже показано содержимое файлов исходных данных модели.

Файл временных задержек переходов (“Tim”) (столбец разделен на 4 части):

1 0.1	19 0.08	0.7	65 95
2 0.02	20 0.2	85 2.5	66 185
3 0.32	21 0.05	86 15	67 495
4 0.02	22 0.2	38 0.5	68 2
5 0.23	23 0.32	39 2	69 7
6 0.32	24 0.2	40 0.7	70 0.8
7 2	25 0.2	41 2.5	71 1.5
8 0.32	26 0.09	42 15	72 15
9 0.32	27 0.64	47 0.5	73 2
10 0.32	28 0.1	48 2	74 5
11 5	29 0.32	49 0.7	75 10
120.14	30 0.2	50 2.5	76 25
130.14	31 0.08	84 15	77 120
140.14	32 0.02	58 3	82 200
15 0.05	33 0	59 6	83 600
16 0.32	36 0.5	60 9	93 1
17 0.09	37 2	61 12	91 2
18 0.32	79	64 35	94 2

Файл начальной разметки (“Vr”).

1000000000000000000000000000000000000000000000000000000000000008
548 44 28 8 19 16 18 9 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Файл приоритетов переходов (“Pri”).

1 31
2 7
3 1
4 7
5 32
6 1
8 1
9 1
10 1
11 4
13 3
126
14 8
155
16 1
17 5
18 1
195
20 1
21 5
22 1
26 5
27 8
28 5

29 1
ЗО 1
31 1
32 32
92 4
90 3
89 2
96 4
95 1

Файл вероятностей выходов ЖГСЧ **(“Gen”).**

52 0.5 0.5
49 0.4 0.3 0.197 0.1 0.003
61 0.01 0.94 0.02 0.03
37 0.2 0.4 0.3 0.1
40 0.2 0.3 0.3 0.2
23 1 0
36 0.98 0.02
35 0.84 0.16
32 0.65 0.3 0.035 0.01 0.005
38 0.75 0.25
26 0.61 0.3 0.035 0.01 0.005 0.04 0
27 0.65 0.3 0.035 0.01 0.005
60 0.97 0.03
24 0.62 0.3 0.05 0.025 0.005
20 0.96 0.04

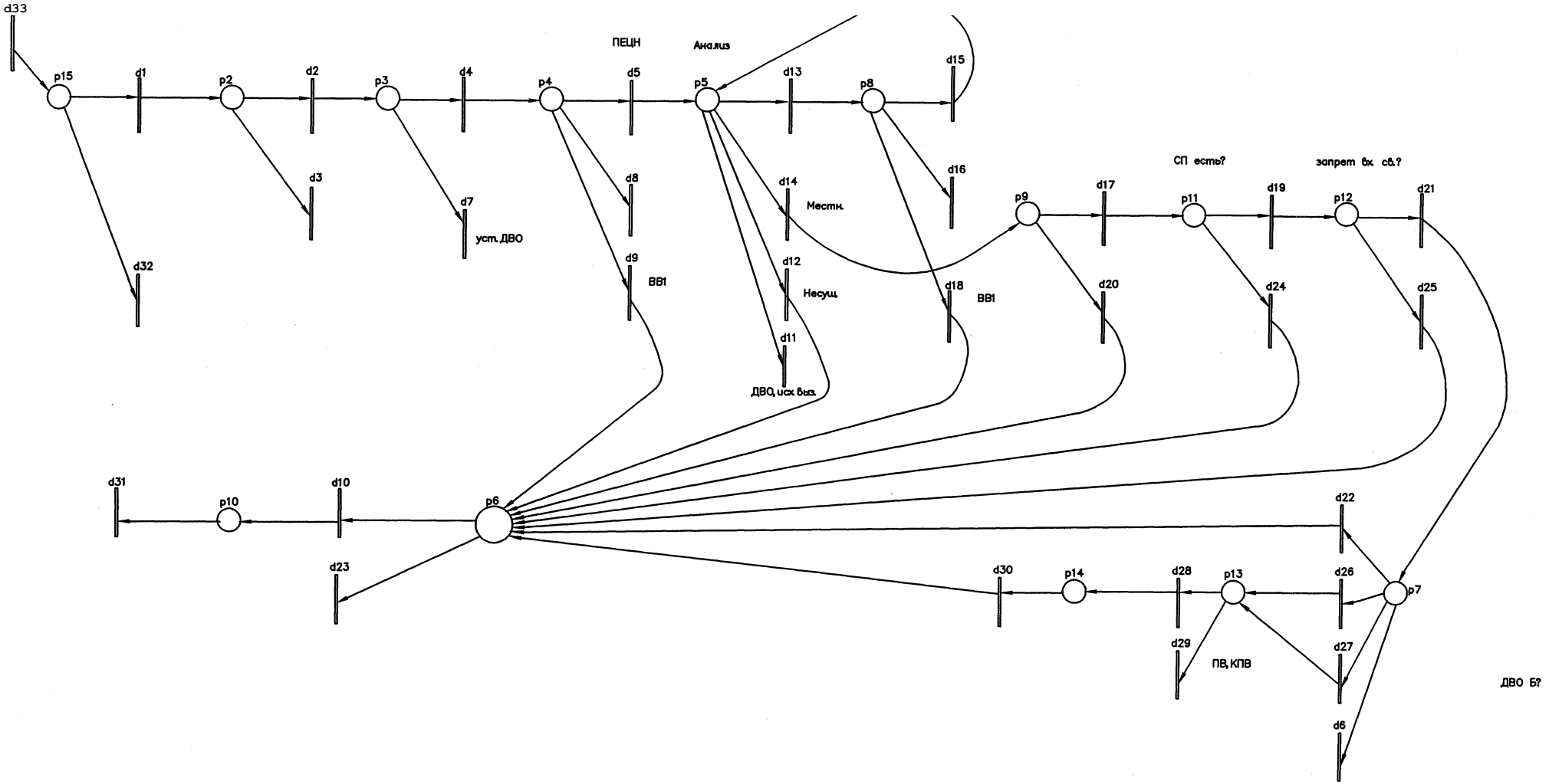


Рис.Д1.3. Поодсеть алгоритма

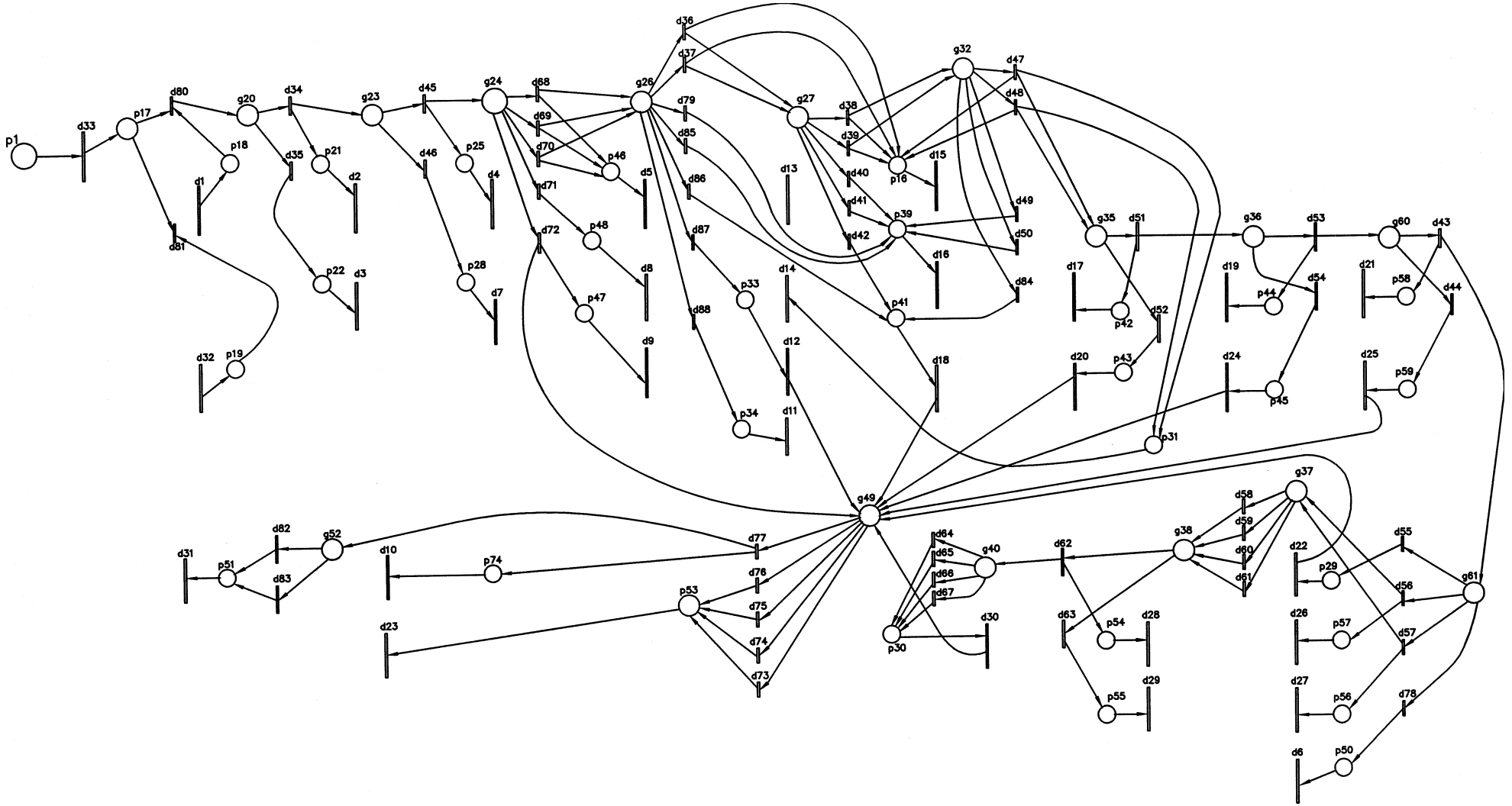


Рис Д 1.4. Поодсеть бнешнего окружения

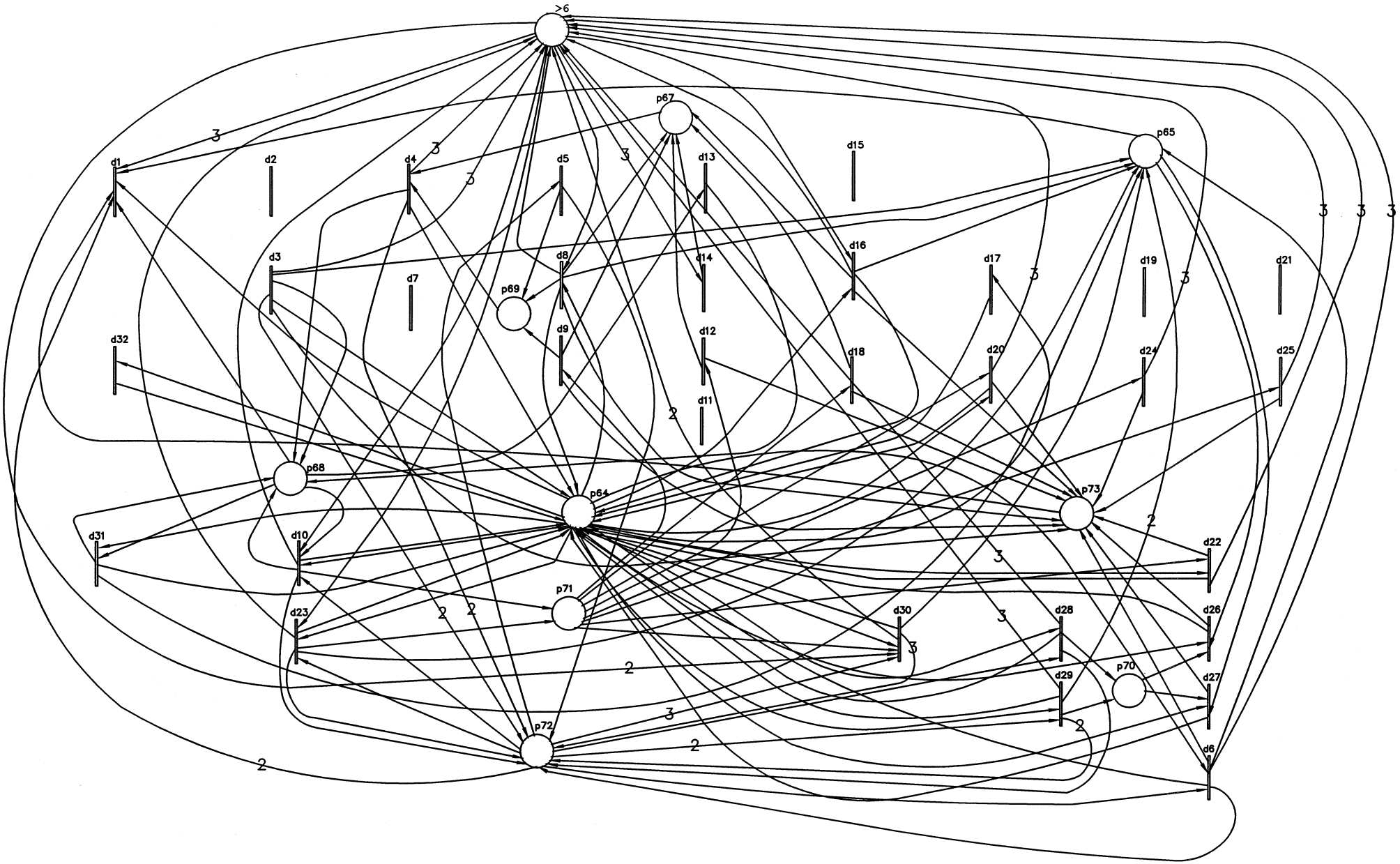


Рис. Д 1.5. Поодсеть ресурсов управления

Д.2. Использование сетевых моделей для оптимизации структуры управления узла коммутации

Как было указано выше, любые методы оптимизации предполагают сопоставление прогнозируемых характеристик возможных реализаций объекта с критериями. Выбор критерия — важная задача, во много определяющая успех оптимизации. Существует много подходов к решению данной задачи [62,68,90]. В некоторых случаях [90] считают, что для оптимизации пригодны лишь экономические критерии, а влияние всех прочих характеристик объекта пересчитывается в экономические показатели. В других случаях считают, что для оптимизации из всего множества характеристик необходимо выделить единственную — определяющую, по которой и ведется оптимизация. Изменения прочих характеристик учитывается в том смысле, что для них устанавливаются ограничения и контролируется нахождение в заданных пределах в процессе оптимизации. Еще одним подходом является учет множества существенных[1] характеристик объекта в виде интегрального показателя. Обычно такой показатель получается применением к данным характеристикам некоторой нелинейной функции вида:

$$K = f(X, X_{min}, y) ,$$

где К — комплексный критерий,

X — множество существенных характеристик объекта,

$X_{ЩП}$ — множество минимальных значений X, по рассматриваемым вариантам,

v— множество нормированных весовых коэффициентов, учитывающих неодинаковую приоритетность и важность характеристик.

Существуют разные рекомендации по выбору функции *f* многочлены, гармонического и показательного типа, произведения сложных функций и прочие, зависящие от некоторым образом нормированных величин — характеристик объекта. Наибольшую неопределенность несет задача выбора весовых коэффициентов, так как требования заказчиков обычно имеют нечеткий, эмпирический характер, и экспертные оценки могут в значительной степени различаться, что снижает степень доверия к ним. Тем не менее данный подход широко применяется, поскольку непосредственный учет различных по-

[1] Существенность понимается в смысле Гельфанда-Цетлина [30] — влияние на численное значение обобщенного критерия.

казателей представляет собой задачу векторной оптимизации, общее решение которой в настоящее время не известно [62].

В работах рассматривающих вопросы оптимизации устройств и систем подобных УК [62,68, и др.] чаще других можно встретить рекомендации по выбору интегрального критерия мультипликативного типа с показательными весовыми коэффициентами вида: *(y* A V V V* *(Y* V»

$$\frac{\text{Л 'm*n A2min л»шiii}}{\{x_{\ell}\}\{x_2\}\text{'''I}\,x_n)}, \qquad (Д.1)$$

$$\begin{cases} \sum_{i=1}^{n} \nu_i = 1 ; \\ \end{cases}$$

(■^imin=o) $^{=,>}$ ($^{1/}$zmin ¯ 0)-

Такой критерий является результатом применения принципа относительной уступки, который целесообразно применять при решении подобных задач [68]. Компромиссом между удобствами критерия и неопределенностью весовых коэффициентов может быть получение семейства кривых $K = f(X,X_m\hat{},\nu)$ при различных значениях $У_г$-. Для определения состава множеств X и $X_m i_n$. Рассмотрим основные характеристики УК, которые разобьем на группы.

1. Характеристики производительности.
   - А. Пропускная способность — максимальная интенсивность обслуженной нагрузки с заданным качеством.
2. Характеристики качества обслуживания.
   - А. Время задержки установления соединения. Складывается из времени ожидания обслуживания $Т_{от}$ и собственно времени установления соединения $Т_у$:

     $Тз = Тот + Ту.$

     С этим показателем связаны так же такие величины, как вероятность превышения нормативного значения $Т_З$ ($Т_З^Н$) и процент времени превышения $Т_З^Н$, а так же средняя длина очереди заявок.
   - В. Вероятность потери вызова $р_П$от.. Складывается из вероятности отказа в установлении соединения по вине УК рот и вероятности установления неправильности соединения или разрушение ранее установленного соединения — неправильного действия УК $р_{нд}$.

3. Технологические характеристики.
   A. Серийноспособность — возможность наращивания оборудования. Характеризуется диапазоном наращивания D_n и шагом наращивания емкости.
   B. Масса, объем, габариты оборудования.
   C. Сложность — обычно количество выводов или корпусов ИС.
   D. Возможности обновления алгоритма работы (например введение новых видов ДВО). Количественно не оценивается.
   E. Потребляемая мощность, тепловыделение.
   F. Технологическая надежность (вероятность отказа по неисправности АО).
4. Экономические характеристики.
   A. Стоимость (общая — УК в целом, удельная — одного номера и т.п.)
   B. Капитальные затраты на производство.
   C. Удельные капитальные затраты (себестоимость одного номера УК и т. п.).
   D. Эксплуатационные расходы.
   E. Приведенные затраты различного типа.
   F. Экономический эффект по отношению к базовому варианту.

Заметим, что данная классификация не претендует на полноту, возможно рассмотрение и других характеристик, в частности специфических для УК, поддерживающих различные режимы коммутации (каналов, пакетов, сообщений). В любом случае количество параметров, по которым могут сопоставляться УК достаточно велико, причем приоритетность требований может меняться, в зависимости от специфики использования УК, экономической ситуации и т.д.

Однозначный выбор в пользу экономических показателей, при котором все прочие пересчитываются в те или иные величины, характеризующие экономический выигрыш (потери), как предложено в [90], в современных условиях нестабильной экономики нецелесообразен. Однако и полный отказ от учета экономических факторов, очевидно привел бы к однобоким, необъективным выводам. В такой ситуации наиболее оптимальным представляется подход, при котором формируется комплексный безразмерный показатель, учитывающий влияние различных характеристик УК. Наиболее обоснованным [62,68,92] по-видимому является интегральный мультипликативный критерий (Д.1).

Число и состав величин, входящих в сомножители выражения, может изменяться в зависимости от конкретного содержания задачи оптимизации и ее размерности.

В данной работе рассматривается вопрос оптимального распределения аппаратных и программных средств в УК в соответствии с некоторым техзаданием, устанавливающим приоритеты характеристик, а не составления самого задания. Поэтому будем считать, что в выражение (Д.1) могут быть подставлены любые заданные величины, а конкретизация полного состава характеристик, по которым будет проводиться оптимизация, здесь не приводится. Тем не менее, можно выделить некоторые основные показатели, которые оказываются существенными практически для любого техзадания. Так для большинства УК, наиболее универсальной характеристикой производительности является пропускная способность УК, важнейшей технологической характеристикой можно считать вероятность отказа АО, а в качестве основной и наиболее универсальной экономической — стоимость в том или ином виде. По определению пропускной способности ясно, что это в свою очередь также комплексный показатель, для подсчета которого необходимо контролировать величины показателей качества функционирования УК, рассмотренные в п.4.2.2.

Таким образом, оптимизацию распределения аппаратных и программных средств в УК можно проводить по указанному критерию, считая основными характеристиками пропускную способность, надежность и стоимость.

Для постановки задачи оптимизации примем следующие формулировки и соглашения. Будем считать ***частным алгоритмом*** совокупность действий УК, описываемых одним переходом подсети алгоритма составленной СП. Совокупность частных алгоритмов, описываемых этой подсетью есть общий алгоритм функционирования УК. Если отдельно рассматривается некоторый набор действий, который строго включается в частный алгоритм, то условимся называть такой набор ***частичным алгоритмом.*** Таким образом общий алгоритм работы ЭУС УК ***U*** является композицией частных ***Ui,*** а последний — композицией частичных ***Uy.***

Будем считать так же, что возможность прогнозирования интенсивности нагрузки и специфика задач, решаемых ЭУС УК определяют способ ***фиксированного*** (статического) распределения задач между отдельными ее УУ [40] (аппаратурными блоками, микропроцессорными модулями и т.п.).

Рассмотрим два основных принципа организации ЭУС УК при фиксированном распределении задач:

1. ЭУС состоит из функциональных блоков (ФБ), за каждым из которых может быть закреплен один или несколько частных алгоритмов, полностью реализуемых данным блоком;
2. ЭУС состоит из функциональных блоков, за которыми могут быть закреплены частичные алгоритмы, входящие в различные частные алгоритмы. Каждый частичный алгоритм обязательно закреплен за одним из ФБ.

В первом случае задача оптимизации может быть сформулирована следующим образом. Заданы наборы функциональных блоков:

$$Ф = \{Фi, Фг,... , Фп\} ,$$

и частных алгоритмов:

$$\mathbf{U= \{ui, U2, ... , Um\}.}$$

Множества Ф и U характеризуются:

вектором-строкой стоимостей функциональных блоков — С*:

$$C^* = |qc_2 \text{ -}C_{„}|, \qquad (Д.2)$$

вектором-строкой вероятностей отказа ФБ — (£:

$$0^* = \backslash<1\backslash <12 \bullet'' <1n\backslash \text{ '} \qquad (Д\text{-}^3)$$

матрицей длительностей выполнения частных алгоритмов ФБ — *Т:*

$$\Gamma = \begin{vmatrix} z_{11} & & hn \\ & \cdots & \\ & & ^mn \end{vmatrix} \qquad (Д\text{-}4)$$

где *n* — число функциональных блоков,

m — количество частных алгоритмов,

Гу — среднее время выполнения j-того частного алгоритма[1] i-м блоком при заданной интенсивности поступления заявок

[1] Включая время инициализации и обмена информацией.

Требуется найти такое закрепление частных алгоритмов за ФБ, при котором обеспечивается экстремум критерия оптимизации, как дискретной функции номера варианта закрепления в некотором упорядоченном множестве всех возможных вариантов.

Во втором случае постановка задачи отличается тем, что заданы средние времена выполнения частичных алгоритмов функциональными блоками в виде клеточной матрицы:

$$T=\begin{array}{|l|l|l|l|l|}
\hline
*111 & 1 & *121 & | & *1и1 \\
*112 & & *122 & I & *1и2 \\
 & | & *12/, & | & ^1«/, \\
\hline
<211 & I & *221 & I & *2и1 \\
*212 & I & *222 & I & *2п2 \\
*21/_2 & | & *22/_2 & | & *2п/_2 \\
\hline
\end{array} \quad ,(4.2.5)$$

^т \ 1	|	^т21	|	^тп \
^т12	|	^т22	|	^тп!
:	|		| :	
	1		1	
*wl/_ra	1	*т2/„	1	^mnl_m

где t_{ijk} — среднее время выполнения k-го частичного алгоритма[1], входящего в j-й частный алгоритм i-м ФБ, при заданной интенсивности поступления заявок;

i=l,2...n, j=l,2...m, k-l,2...lj', n, m, lj —натуральные числа.

Если частичные алгоритмы определены так, что при обращении к включающему их частному алгоритму обязательно выполняются все частичные, причем только один раз, то остальные исходные данные такие же, как и в первой постановке. В противном случае каждый частный алгоритм должен быть охарактеризован вектором среднего числа выполнения частичных алгоритмов *f j:*

$$Л = fД'-'$$

где *j* — номер частного алгоритма,/ = *1, 2,..., m ,*

1_m — количество частичных алгоритмов в m-ом частном,

[1] Включая время инициализации и обмена информацией.

fjim — среднее число выполнений 1_m-*го* частичного алгоритма при однократном выполнении j-ого частного алгоритма.

Возвратимся к первой постановке задачи. Определим общее число всевозможных вариантов размещения *m* алгоритмов в п ФБ. Таблицу размещения алгоритмов размера *m* x *n* можно рассматривать, как и-ичный m-разрядный счетчик (см. пример на рис. Д.2.1), число состояний которого:

$$X = n^m. \qquad (Д.6)$$

Варианты, предполагающие размещение всех алгоритмов в одном ФБ, соответствуют либо централизованным ЭУС, либо полной однотипности всех ФБ.

	$Ф\backslash$ $Ф_2$	$Ф_3$	$Ф^\wedge$
u_x	x			
u_2		x		
u_3				x
u_4			x	
u_5		x		

Рис.Д.2.1. Пример таблицы размещения алгоритмов

При числе алгоритмов т=30 и номенклатуре из четырех ФБ (п=4) общее число всевозможных вариантов составит $X=4^{30}$, то есть более чем 10^{18}. Ясно, что для таких больших чисел, даже при использовании наилучших современных компьютеров, невозможно реализовать выполнение каких-либо операций. Поэтому на первом этапе ставится задача уменьшения числа рассматриваемых вариантов до такой величины, которая бы позволила бы хотя бы грубо, на основании простейших вычислений, сопоставить их между собой и выбрать некоторое подмножество детально рассматриваемых структур. Для такого ограничения, очевидно, пригодны лишь методы эвристического анализа.

Возможным подходом может быть уменьшение показателя степени m в выражении (Д.6) путем объединения некоторых частных алгоритмов в группы, которые должны выполняться в одном (любом) ФБ. Принцип такого объединения — выделение более-менее обособленных, функционально тесно связанных множеств алгоритмов из общего их числа на основе анализа их роли и места а общем алгоритме ЭУС, степени взаимосвязи между собой, направленности на управление одним объектом, частоте обращений к одним и тем же массивам памяти и т.д. Примеры подобного объединения ча-

стных алгоритмов можно наблюдать во многих практических реализациях ЭУС. В частности, в рассмотренных в разделе 2.3 (рис.2.32) цифровых УК типа DX-200 (ЭАТС-200) (Финляндия) [77], управляющие программы, реализующие различные частные алгоритмы объединены в процессы обслуживания вызовов:

- SUBSIG — абонентской сигнализации,
- CASSIG — линейной сигнализации,
- INREGI — внутренней регистровой сигнализации,
- DUREGI — внешней регистровой сигнализации,
- SWICOP — управления коммутацией (в КП ГИ), и другие.

Каждый процесс реализован программами одного ФБ (с некоторыми исключениями) — SSU, LSU, RU, M и др. [12,77].

Число групп алгоритмов, получившихся в результате объединения должно быть достаточно малым, настолько, чтобы выполнение числа пустых циклов, определяемого выражением (Д.6) занимало не более нескольких минут на имеющемся компьютере. В частности, для популярного, в настоящее время, персонального компьютера Pentium-266 с процессором типа AuthenticAMD Am586 (Кб) Модель 7 (встроенный сопроцессор) и рабочей частотой 266 МГц, при использовании компилятора Turbopascal 5.0 такое число ориентировочно можно принять не большим 2^{30}.

Кроме группировки алгоритмов, уменьшения размерности задачи можно достичь, если, исходя из особенностей рассматриваемых ФБ и частных алгоритмов, предварительно закрепить некоторые алгоритмы за определенными ФБ или группами ФБ. Так, за наиболее дешевыми блоками можно закрепить более простые, но часто исполняемые алгоритмы (например контроль состояния АЛ). Если предполагается аппаратная реализация некоторого ФБ, то возможно предварительное закрепление алгоритмов критичных к скорости исполнения и не подверженных модификациям в процессе эксплуатации и совершенствования УК.

После снижения размерности задачи до приемлемой для простейшего анализа величины необходимо выбрать процедуру сопоставления вариантов при переборе. Основные требования к такой процедуре — малое время вычислений по каждому варианту и такая точность оценки, которая достаточна для того, чтобы не потерять оптимальный вариант.

Возможный подход к решению данной проблемы можно получить на основе следующих соображений. Как было указано в разделе 4.1, наиболее общими и универсальными требованиями к разрабатываемым устройствам, подобным УК, являются их производительность, стоимость и надежность, которые характеризуют, соответственно технический, экономический и качественный аспекты системы. Ясно, что эти показатели системы зависят от соответствующих показателей ее компонентов, степени участия компонент в общем технологическом процессе и характера их взаимодействия между собой. Поэтому на втором этапе ограничения множества детально исследуемых структур предлагается охарактеризовать факт закрепления некоторого частного алгоритма за определенным ФБ — обобщенной предварительной оценкой, учитывающей среднее время выполнения алгоритма *ty* (Д.4), стоимость ФБ , C_L (Д.2), вероятность его отказа q_t (Д.3), а так же среднюю частоту выполнения частотного алгоритма *Wj,* при условии, что ограниченность ресурсов УК не оказывает влияния на обслуживание вызовов (идеальный безотказный УК). Первые три величины содержатся в исходных данных задачи оптимизации, а последнюю можно легко получить путем испытаний сетевой модели УК с исключенной подсетью ресурсов.

Как было указано в разделе 4.1, для получения обобщенной оценки по нескольким показателям необходимо назначить весовые коэффициенты v , характеризующие их приоритетность (4.1). Связь между первыми тремя показателями , отображающими основные требования УУ предполагается установить в соответствии с (4.1), а учет частоты выполнения частных алгоритмов — простым умножением. Тогда для каждой клетки таблицы алгоритмов (рис.Д.2.1) получим величину промежуточного критерия *к;..* ′■

$$k'_{ij} = \left(\frac{t_{ij}}{t_{\max}}\right)^{v_t} \cdot \left(\frac{c_{\min}}{c_i}\right)^{v_c} \cdot \left(\frac{q_{\min}}{q_i}\right)^{v_q} \cdot W_j \ ,$$

где *ty, Ci* ,qj — компоненты выражений (Д.3), (Д.2),(Д.6) соответственно,

tmax —максимальный элемент матрицы Т:

tmax Шах J ,

Cmtn— минимальная стоимость ФБ :

$C_{wz,,}$=min {C_z-},

q_{min} — вероятность отказа самого надежного ФБ :

Wj — средняя частота выполнения j -го частного алгоритма по результатам испытаний “идеального” УК,

v_c, V_t, v_q — весовые коэффициенты по стоимости, производительности и надежности соответственно:

$$v_c + v_t + v_q = l .$$

После предварительного определения промежуточных критериев можно приступить к перебору вариантов закрепления частных алгоритмов и сопоставления их по суммарному критерию *ку* . Таким образом, в цикле по большому числу вариантов необходимо реализовать только относительно простые операции сложения и сравнения. Ясно, что такая упрощенная оценка различных вариантов не гарантирует отыскание оптимального, так как не учитывает характера взаимодействия алгоритмов, влияния ограниченности ресурса. Однако она позволяет ориентировочно ранжировать их по близости к оптимуму с тем, чтобы сосредоточить внимание на сравнительно небольшом множестве наилучших, по предварительной оценке, структур.

Некоторую неопределенность несет в себе вопрос о количестве наилучших вариантов, принимаемых к детальному рассмотрению, поскольку трудно указать какую-либо четкую границу, в пределах которой гарантируется нахождение оптимума. Невидимому следует стремиться к возможно большему числу вариантов, которое позволяют проанализировать имеющиеся в распоряжении разработчика вычислительные ресурсы за отведенное для моделирования время.

На рис.Д.2.2 показан возможный алгоритм ограничения множества детально исследуемых структур (массив R). Обозначение S — номер возможного варианта, отождествляемый с состоянием и-ичного /«-разрядного счетчика таблицы вариантов (рис.Д.2.1). Принцип работы алгоритма — сравнение очередного варианта с наихудшим из ранее выбранных и если он лучше — вставка его номера в упорядоченный массив R. Место вставки можно определять известным методом бинарных сравнений (рис.Д.2.2.б) или другим. В результате выполнения алгоритма в массиве R формируются номера S и значения суммарного критерия *Ък’* по V условно наилучшим вариантам структур.

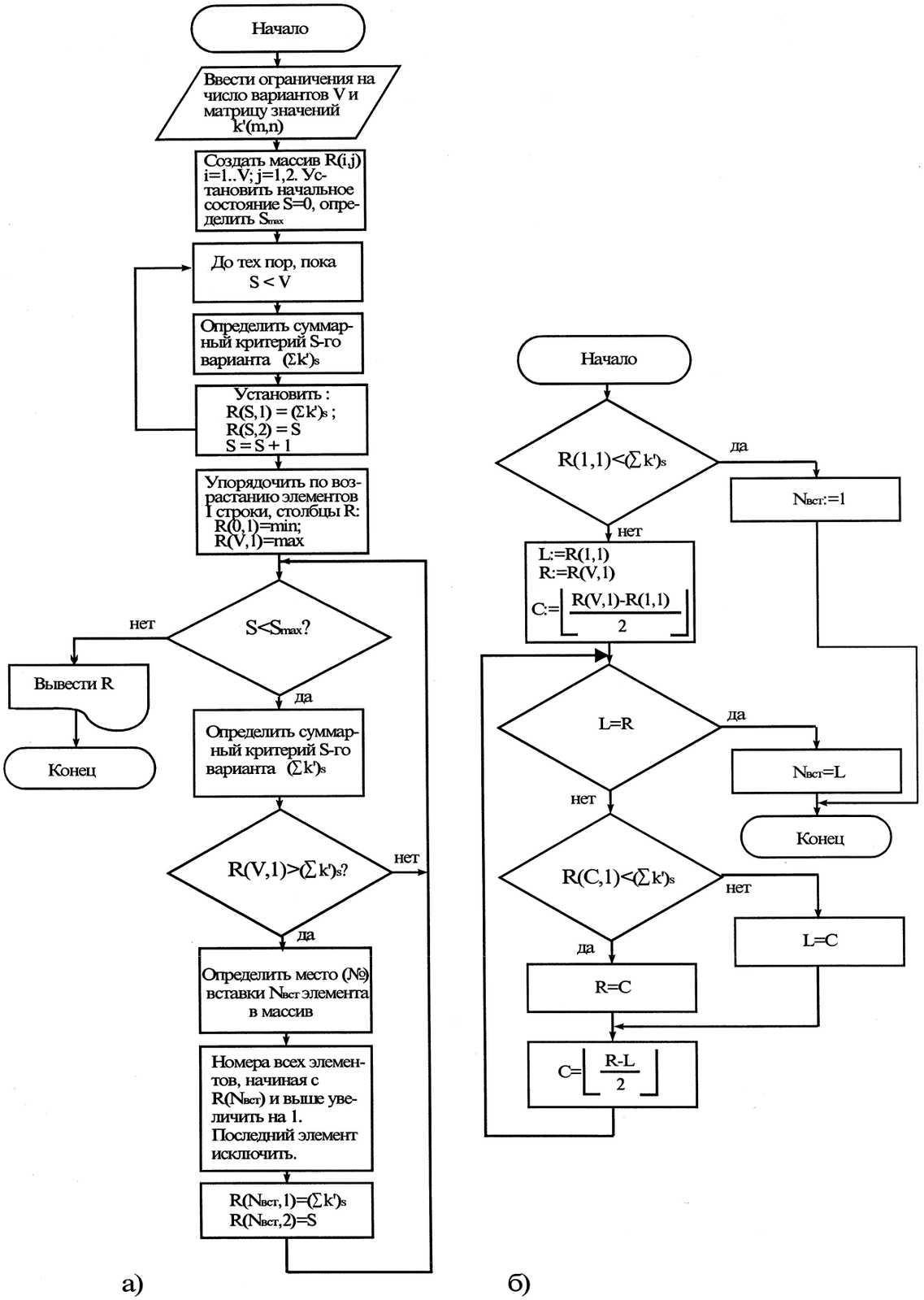


Рис.Д.2.2. а) Общий алгоритм ограничения вариантов структур,

б) алгоритм определения места вставки NВСТ элемента в массив

На третьем этапе необходимо определить конкретные показатели функционирования УК и общий интегральный критерий по каждому из детально исследуемых вариантов с учетом ограниченности ресурсов, взаимодействия частных алгоритмов, вероятностных характеристик внешней среды. Каждый вариант структуры из полученного на втором этапе множества детально исследуемых вариантов задает подсеть ресурсов и временные задержки переходов подсети алгоритма. Таким образом, детальное исследование полученного множества структур состоит в :

- доопределении подсетей алгоритма и ресурсов общей сетевой модели УК,
- дополнении ее подсетью анализа, обеспечивающей вычисление интересующих показателей функционирования УК,
- организации испытания модели,
- вычислении обобщенных критериев по каждому варианту по результатам испытаний,
- сравнении результатов и выборе наилучшей структуры.

Таким образом, на третьем этапе оптимизации, действия по выбору оптимальной структуры определяются изложенной в работе методикой построения сетевых моделей.

В случае постановки задачи оптимизации, соответствующей второму принципу организации ЭУС УК, для ограничения числа вариантов на первом этапе необходимо иметь таблицу размещения алгоритмов такого же типа как на рис.Д.2.1. Это можно получить, если строкам таблицы сопоставить связанные функционально наборы частичных алгоритмов, входящие в любые частные. Такая операция эквивалентна перераспределению положения строк матрицы (Д.5) таким образом, что они группируются по признаку закрепления соответствующих частичных алгоритмов за общим ФБ. В этом случае в основном сохраняется применимость описанного метода оптимизации. Отличие заключается в предварительном определении промежуточного критерия *k-j,* так как в этом случае каждой клетке таблицы размещения алгоритмов будет соответствовать уже обобщенный по группе критерий, взвешенный по частоте исполнения отдельных частичных алгоритмов. Для такого обобщения можно, например воспользоваться выражением :

$$\text{Ділах} > \frac{c_{min}}{\#_{min}} \quad q \qquad \text{(Д.8)}$$

В данной формуле суммирование предполагается по выделенной группе частных алгоритмов. Заметим, что в любой постановке задачи целесообразно произвести нормирование *к'у* для уменьшения объема вычислений при переборе.

Описанный метод оптимизации структуры иллюстрирует применимость предлагаемой методики моделирования к задачам разработки УУ УК.