

Гофман Е. А., Олейник А. А., Субботин С. А.
СОКРАЩЕНИЕ БАЗ ЛИНГВИСТИЧЕСКИХ ПРАВИЛ
НА ОСНОВЕ ДЕРЕВЬЕВ РЕШЕНИЙ

Рассмотрена задача индукции лингвистических правил. Разработан метод идентификации деревьев решений для индукции лингвистических правил. Создано программное обеспечение на основе предложенного метода. Проведены эксперименты по решению практических задач, что позволило исследовать эффективность предложенного метода.

Ключевые слова: дерево решений, индукция правил, лингвистическое правило.

УДК 004.78; 004.021; 004.046

Gofman Ye., Oliynyk A., Subbotin S.
LINGUISTIC RULES BASES REDUCTION BASED ON
DECISION TREES

The problem of linguistic rules induction is considered. A method of decision trees identification for linguistic rules induction is developed. The software based on the proposed method is created. Experiments on the solution of practical problems, which allowed to investigate the effectiveness of the proposed method are made.

Key words: decision tree, rules induction, linguistic rule.

Ильяшенко М. Б.¹, Голдобин А. А.²

¹Канд. техн. наук, доцент Запорожского национального технического университета

²Ассистент Запорожского национального технического университета

РЕШЕНИЕ ЗАДАЧИ ПОИСКА ИЗОМОРФИЗМА ГРАФОВ ДЛЯ ПРОЕКТИРОВАНИЯ СПЕЦИАЛИЗИРОВАННЫХ ВЫЧИСЛИТЕЛЕЙ

Предлагается усовершенствованный алгоритм поиска изоморфизма графов и результаты исследования его эффективности. Объектом исследования является множество граф-схем алгоритмов достижения цели, полученная после обхода заданной семантической сети абстрактной машиной Уоррена.

Ключевые слова: декларативная логика, предикат, дерево вывода, пролог, рекурсивный обход с возвратом, граф-подграф изоморфизм.

ВВЕДЕНИЕ

В составе программно-лингвистических средств автоматизации проектирования цифровых устройств на программируемых логических интегральных схемах (ПЛИС) широко используются методики формального описания структурно-функциональной организации проектируемого объекта. Методология проектирования на основе математического аппарата теории ориентированных гиперграфов [1] позволяет использовать унифицированные алгоритмы выполнения основных этапов создания цифровых устройств, в том числе специализированных (проблемно-ориентированных) вычислителей.

Ориентированный гиперграф, допускающий петли и кратные дуги, является наиболее общим типом графовых моделей и называется ориентированным псевдогиперграфом. В дальнейшем, для краткости, ориентированный, помеченный псевдогиперграф будем именовать «графом».

Для формального описания свойств узлов и дуг графа удобно применять операторное пространство, образующее многозначную логику. Примером такого пространства является трехзначная логика модальных операторов Лукасевича [2].

Применение многозначного операторного пространства для описания актов синтеза цифрового устройства, которое формально задано графом, и исследование способов преобразования проблемно-ориентированных описаний является актуальной задачей. Ее решение позволяет разрабатывать эффективные инструменты, мес-

то применения которых – системы автоматизированного проектирования (САПР) цифровых и микропроцессорных устройств различного назначения.

Абстрактная машина Уоррена (англ. Warren's abstract machine, WAM) [3] представляет собой формальную модель устройства, реализующего основные операции исчисления предикатов первого порядка, представленных дизъюнктами Хорна. Известное приложение WAM – японский проект вычислителей пятого поколения (1982–1992 гг.).

В абстрактной машине Уоррена одновременно выполняется семантический анализ программы на входном языке L_{pro} и формирование таблиц лексико-синтаксического анализа. Результаты анализа используются для генерирования программы на выходном языке L_{wam} . Язык L_{wam} представляет собой набор типовых, функционально-ориентированных операций линейной резолюции в терминах WAM – конечного множества команд типа *unify_variable*, *put_list* и др.

Формальная грамматика является абстрактной структурой, описывающей множество правил образования строк языка из заданного алфавита терминальных и нетерминальных символов. Генерирующая и анализирующая формальные грамматики используются для решения противоположных задач, в зависимости от семантики грамматического разбора. Генерирующая грамматика образуется конечным множеством порождающих правил (продукций) формирования строк формального

языка. Анализирующая грамматика предназначена для грамматического разбора строк формального языка, поступающих на вход.

Графически правила продукций формальной грамматики представляются в виде ориентированных помеченных псевдогиперграфов. Например, продукции регулярной грамматики описываются в виде *граф-схем алгоритма* (ГСА).

Иерархия Хомского [2] представляет собой иерархию классов последовательно вложенных формальных грамматик, описывающих языки разного типа. В исходном виде иерархия Хомского образована четырьмя уровнями, которые задают законы образования основных классов формальных языков. Каждому уровню иерархии Хомского соответствует своя модель распознающего автомата, начиная с машины Тьюринга с бесконечной лентой, которая предназначена для распознавания продукций грамматик общего вида (уровень 0 иерархии Хомского), и заканчивая конечным автоматом, порядок переключения состояний которого описывается регулярной грамматикой (уровень 3 иерархии Хомского).

Абстрактная машина Уоррена способна распознавать грамматики уровня 2. Это контекстно-свободные грамматики, с помощью которых определяются контекстно-свободные языки. Языки, образованные такими грамматиками, могут быть распознаны с помощью *стекового автомата*. Таким образом, с точки зрения анализа входной программы, поведение WAM эквивалентна работе стекового автомата.

В процессе выполнения запроса WAM способна генерировать описание ориентированного графа, представляющего ГСА выходной программы на регулярном языке L_{wam} .

Так как абстрактная машина Уоррена занимает в иерархии Хомского более высокий уровень по отношению к конечному автомату и способна преобразовывать строки контекстно-свободного языка L_{pro} в конструкции регулярного языка L_{wam} , она может использоваться в следующих приложениях:

- ввод задания на проектирование специализированного вычислителя на ПЛИС;
- конфигурирование микропроцессорных систем с программируемой архитектурой;
- реализация технологий логического, концептуального, функционального программирования.

В данной работе описан алгоритм поиска граф-подграф изоморфизма и приведены результаты исследования его эффективности на примере помеченных ориентированных гиперграфов, которые представляют результат работы абстрактной машины Уоррена.

1. ПОСТАНОВКА ЗАДАЧИ

Графически концептуальная область входной программы для абстрактной машины Уоррена может быть задана *семантической сетью*. Например, на рис. 1 показан концептуальный граф вычислительной модели прямоугольного треугольника.

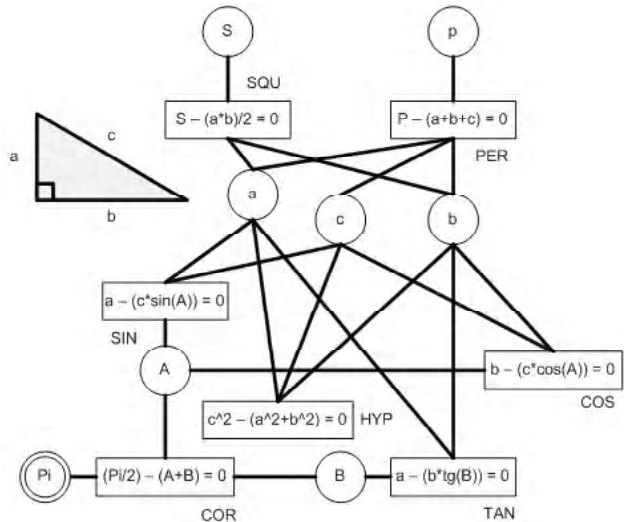


Рис. 1. Концептуальный граф

Программа на входном языке L_{pro} абстрактной машины Уоррена образована объектами двух типов: термом программы p и термом запроса $?-q$. Цель работы машины можно сформулировать так: определив программу p , необходимо составить запрос $?-q$, выполнение которого или закончится неудачно, если p и q нельзя унифицировать, или будет завершено удачно, если получилась связка переменных из q с переменными из p .

При доказательстве теорем методом резолюции, проверка невыполнимости запроса наталкивается на препятствия, связанные с бесконечным числом областей интерпретации запроса. В общем случае, если выбранная область бесконечна, то запрос допускает бесконечно много конкретизаций, т. е. существует бесконечно много интерпретаций относительно языка L_{pro} . Для обхода данной проблемы в WAM клаузная форма концептуального пространства, заданная программой p , ограничивается эрбановой областью [2].

По этой причине пролог-процессор, реализованный на WAM, реализует стратегию разбора И/ИЛИ-дерева запроса «слева направо и вглубь с возвратом при неудаче». На рис. 2 показан случай поиска конкретизации переменной b в вычислительной модели прямоугольного треугольника, при условии, что грамматический разбор начинается с фразы SQU.

Можно сделать выводы относительно порядка работы машины Уоррена.

1. Концептуальное пространство, в котором выполняется грамматический разбор запроса $?-q$ машиной Уоррена ограничивается множеством конкретизированных во входной программе p переменных и функционалов.
2. Время выполнения запроса $?-q$ зависит от выбора начальной языковой конструкции – хорновского дизъюнкта, определяющего правило в программе на входном языке L_{pro} .
3. Если $H_M^k(G)$ – множество всех ГСА, сформированных машиной Уоррена при выполнении k -го запроса на максимально определенном пространстве, то лю-

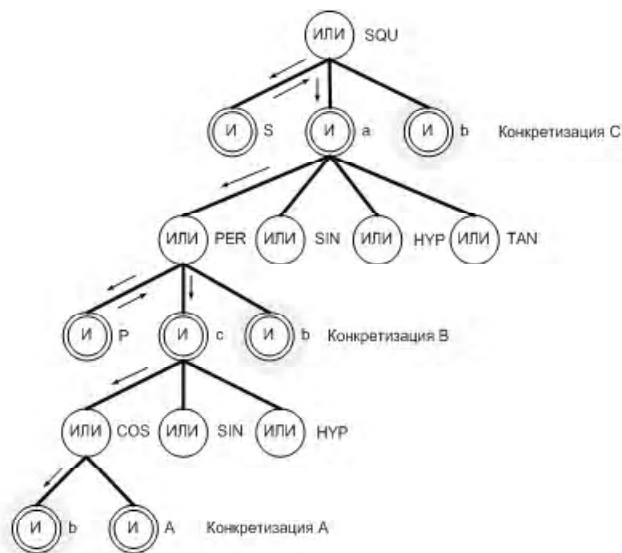


Рис. 2. Пример грамматического разбора

бое множество $H_m^k(G)$ не полностью определенных концептуальных пространств сюръективно отображается на множество $H_M^k(G)$: $H_M^k(G) \subseteq \forall H_m^k(G)$.

Не полностью определенным концептуальным пространствам соответствуют различные семантические сети, представляющие их графически. Поиск изоморфного подграфа в ориентированном графе семантической сети концептуального пространства программы для WAM является важной задачей, обеспечивающей значительное повышение эффективности работы генератора программно-лингвистических описаний, созданного на основе машины Уоррена. Для решения этой задачи был разработан усовершенствованный алгоритм определения изоморфизма двух заданных графов.

2. МЕТОД РЕШЕНИЯ ЗАДАЧИ

Пусть даны графы $G_1 = (V_1, E_1, L_1)$ и $G_2 = (V_2, E_2, L_2)$, где V – множество вершин графа, E – множество ребер графа и L метки вершин графа. Граф G_1 изоморфен подграфу графа G_2 (обозначается, как $G_1 \cong S_2 \subseteq G_2$), если существует подстановка $\phi: V_2 \rightarrow V_1$, такая, что для каждой пары вершин $v_i, v_j \in V_2$, если $(v_i, v_j) \in E_2$, то $(\phi(v_i), \phi(v_j)) \in E_1$ и для всех $l_i \in L_2$ выполняется $l_i \in L_2 = \phi(l_i) \in L_1$.

Алгоритм установления граф-подграф изоморфизма для помеченных графов является развитием и продолжением алгоритма установления изоморфности [4].

Алгоритм установления изоморфизма удобно описывать в терминах поиска в пространстве состояний. Каждое состояние s процесса совмещения вершин соответствует частичной подстановке $\phi(s)$, которая содержит часть вершин полной подстановки. Каждому состоянию так же соответствуют подграфы $G_1(s)$ и $G_2(s)$, полученные из вершин графов G_1 и G_2 , вошедших в

частичную подстановку $\phi(s)$, и ребер, соединяющих эти вершины. В дальнейшем обозначим через $\phi_1(s)$ и $\phi_2(s)$ проекции подстановки $\phi(s)$ на V_1 и V_2 .

Алгоритм состоит из предварительной и основной части. В предварительной части выполняются операции упорядочивания вершин графов и выполнения однократных, по ходу алгоритма, операций, призванных сократить область поиска основной, переборной части алгоритма.

Предварительная часть алгоритма. Основные действия, выполняемые в предварительной части алгоритма – сортировка вершин графов и формирование матрицы возможных совмещений.

Матрица возможных совмещений $M_{i,j}$ – это бинарная таблица размером $|V_1| \times |V_2|$. Каждому элементу таблицы соответствует пара вершин исходных графов $V_{1,i}$ и $V_{2,j}$. Значения матрицы формируются следующим образом:

- $M_{i,j} = 0$, если на основании предварительных проверок вершины $V_{1,i}$ и $V_{2,j}$ совместить нельзя;
- $M_{i,j} = 1$, в противном случае.

Смысл матрицы возможных совмещений в том, чтобы выполнить однократно в рамках предварительной части алгоритма все проверки, не основанные на информации, полученной в процессе совмещения вершин, тем самым, ускорить обработку соответствующих ограничений, сведя ее к одной операции сравнения.

В программе реализованы следующие предварительные проверки:

1. $M_{i,j} = 0$, если $|V_{1,i}| < |V_{2,j}|$, где $|V_{X,Y}|$ – степень вершины Y графа X ;
2. $M_{i,j} = 0$, если $|V_{1,i}^{in}| < |V_{2,j}^{in}|$, где $|V_{X,Y}^{in}|$ – число входящих ребер вершины Y графа X ;
3. $M_{i,j} = 0$, если $|V_{1,i}^{out}| < |V_{2,j}^{out}|$, где $|V_{X,Y}^{out}|$ – число исходящих ребер вершины Y графа X ;
4. $M_{i,j} = 0$, если $W_{1,i}^{vertex} < W_{2,j}^{vertex}$, где $W_{X,Y}^{vertex}$ – число вершин в волновом разложении подграфа окружения вершины Y графа X ;

$$5. \quad M_{i,j} = 0, \quad \text{если} \quad \sum_{l=1}^k W_{1,i,l}^{vertex} < \sum_{l=1}^k W_{2,j,l}^{vertex},$$

$k = 1..|W_{2,j}^{vertex}|, l = 1..4$, где $W_{X,Y,l}^{vertex}$ – число вершин в l -ой волне волнового разложения графа X , начиная с вершины Y ;

6. $M_{i,j} = 0$, если $W_{1,i}^{ribes} < W_{2,j}^{ribes}$, где $W_{X,Y}^{ribes}$ – число ребер в волновом разложении подграфа окружения вершины Y графа X ;

$$7. \quad M_{i,j} = 0, \quad \text{если} \quad \sum_{l=1}^k W_{1,i,l}^{ribes} < \sum_{l=1}^k W_{2,j,l}^{ribes},$$

$k = 1..|W_{2,j}^{ribes}|, l = 1..4$, где $W_{X,Y,l}^{ribes}$ – число ребер в l -ой волне волнового разложения графа X , начиная с вершины Y ;

8. $M_{i,j} = 0$, если $L_{1,i} \neq L_{2,j}$, где $L_{X,Y}$ – метка вершины Y графа X .

Возможно использование и других критериев для оценки возможности совмещения вершин графов. Метод разработки таких критериев основан на волновом разложении графов, начиная с заданной вершины [5]. По мере распространения волны получают подграфы окружения вершин. Сравнивая параметры соответствующих подграфов окружения вершин графов, которые предполагается совмещать, делается вывод о потенциальной возможности или принципиальной невозможности такого совмещения. В приведенных критериях для этого использовались сумма вершин и ребер в подграфах окружения сравниваемых вершин для всех этапов распространения волны.

Сортировка вершин графов производится с целью ускорения нахождения изоморфной подстановки, в случае, если такая подстановка существует. В переборной части алгоритма переставляются только вершины большего графа, в то время, как порядок вершин меньшего графа не меняется. Порядок следования вершин меньшего графа определяется в предварительной части алгоритма.

Пусть $T_{2,i}$ – количество ребер инцидентных вершинам с меньшими номерами и $P_{2,i} = \sum_{j=1}^{|V_1|} M_{j,i}$ – суммарное количество вариантов совмещения вершины i графа G_2 с вершинами графа G_1 . Тогда порядок сортировки вершин графа G_2 следующий:

$$V_{2,i} = V_{2,k}, \text{ где } T_{2,k} = \min_{j=i+1}^{|V_2|} (T_{2,j}).$$

Если $T_{2,i} = T_{2,j}$, то $V_{2,i} = V_{2,k}$, где $P_{2,k} = \min(P_{2,i}, P_{2,j})$.

Т. е. вершины графа G_2 сортируются в порядке убывания количества связей с вершинами имеющими меньшие номера или в порядке убывания количества вариантов совмещения вершин, если количество связей одинаково. Такой порядок следования вершин обусловлен тем, что чем больше связей с уже совмещенными имеет вершина, тем жестче будет ограничивающее условие, включающее эту вершину, и, соответственно, меньше общее количество совмещений, которые необходимо перебрать.

Основная часть алгоритма. Эта часть алгоритма представляет собой последовательное наложение вершин с возвратом, описывать которое удобно в терминах метода поиска в пространстве состояний.

Вершины графа G_2 остаются нетронутыми и каждой из них ставится в соответствие одна из вершин графа G_1 . При этом проверяется допустимость такого совмещения. Если удастся найти соответствие всем вершинам графа G_2 , при этом выполнено условие изоморфизма, то найденное состояние возвращается как искомая подстановка.

Пусть $T_{1,i}$ – количество связей вершины i графа G_1 с вершинами $V_{1,j} \in \Phi_1(s)$, а $T_{2,i}$ – количество связей вершины i графа G_2 с вершинами $V_{2,j} \in \Phi_2(s)$.

Начальному состоянию $\Phi(s)_0 = 0$ соответствует состояние, при котором не совмещено еще ни одной пары вершин.

Для получения i -го состояния для вершины $V_{2,i}$ ищется соответствие среди вершин $V_{1,j}$, таких что:

1. $M_{i,j} = 1$, т. е. вершины совместимы на основании предварительных проверок;
2. $T_{1,i} \geq T_{2,j}$;
3. Для $k = 1..i$, если $(v_i, v_k) \in E_1$, то $(\Phi(v_i), \Phi(v_k)) \in E_2$.

Если выполнены все три условия, из которых третье является прямым следствием определения граф-подграф изоморфизма, то соответствующая пара вершин входит в частичную подстановку и формируется новое состояние $\Phi(s)_i$.

Перебор состояний производится методом поиска в глубину.

3. РЕЗУЛЬТАТЫ РАБОТЫ

Результат работы машины Уоррена может быть представлен в виде ГСА – графовой модели, в которой используются следующие типы вершин (блоков) (рис. 3).

Для тестирования производительности разработанного алгоритма формировались случайные графы, состоящие из блоков P и D типа. При формировании графов использовались следующие параметры:

- nv – число вершин в генерируемых графах;
- ns – нижняя граница число вершин в генерируемых подграфах;
- p – процент вхождения P -блоков (процент вхождения D -блоков составлял $100\% - Np$);
- l – количество различных меток, используемых для маркировки вершин графов.

ГСА формировался из отдельных блоков P и D типа, на основании параметра p в соотношении $p/(100-p)$. Дуги блоков случайным образом соединялись между собой. Общее число вершин ГСА определялось параметром nv . Всем вершинам случайным образом приписывались метки в диапазоне от 0 до $l-1$.

Подграф формировался путем удаления части вершин из ГСА. Начиная со случайной вершины, пускалась волна, проходящая как по входящим, так и по исходя-

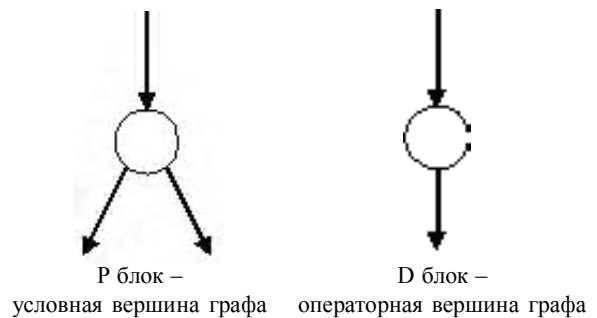


Рис. 3. Типы вершин графа

щим дугам, для формирования связного подграфа. Как только на очередном шаге размер подграфа вошедшего в волновое разложение превышал параметр ns , все вершины, которые накрыла волна, выделялись в виде подграфа.

Приводятся результаты численного исследования производительности алгоритма на основании набора сгенерированных графов, описанного выше. Каждое значение формировалось как суммарное время поиска подграфа для 100 пар графов (рис. 4).

Для графов с числом вершин до 1500 включительно, алгоритм в среднем тратит на поиск изоморфного подграфа не более 0,35 секунды машинного времени, что достаточно для решения реальных задач проектирования схемных устройств управления. Ввиду комбинаторной природы алгоритма, наблюдаются пики производительности, при которых время поиска подграфа в отдельных случаях может заметно отличаться от среднего (рис. 5).

Влияние размера искомого подграфа на общую производительность алгоритма значительно менее весомое. При изменении размера подграфа по отношению к размерам ГСА от 1 % до 50 % (т. е. в 50 раз) время работы алгоритма изменилось лишь с 28 секунд до 100 (т. е. в 3 раза). С учетом

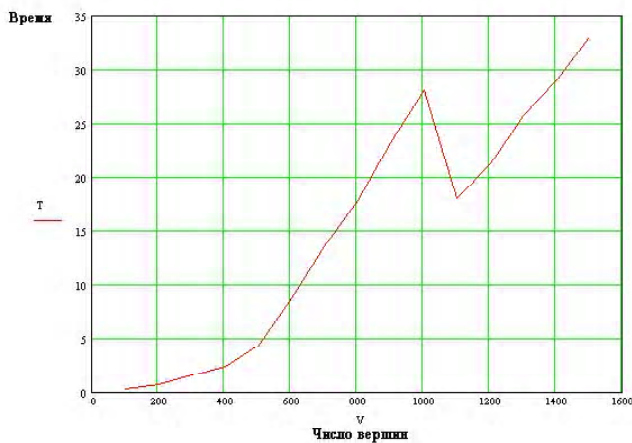


Рис. 4. Зависимость времени поиска подграфа от числа вершин в графе конечного автомата (параметры $ns=20$, $p=20$, $l=5$, $nv=100\dots1500$)

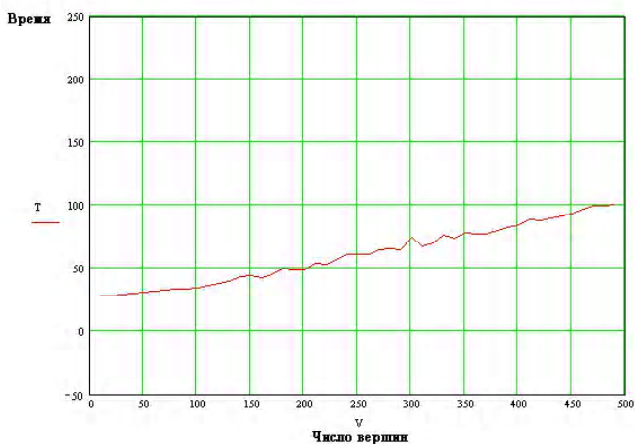


Рис. 5. Зависимость времени поиска подграфа от числа вершин в подграфе типового ГСА (параметры $nv=1000$, $p=20$, $l=5$, $ns=10\dots500$)

того, что большинство ГСА функционально-ориентированных конечных автоматов имеют размер порядка 20–40 вершин, фактор размера подграфа не является решающим для производительности алгоритма (рис. 6).

При изменении процента блоков P -типа использованных для генерации графов конечного автомата в диапазоне от 10 % до 50 % (в 5 раз), время вычислений изменилось с 33 до 50 секунд (т. е. в 1,5 раза). Следовательно, как и фактор размера искомого подграфа, соотношение числа блоков P и D типов незначительно влияет на производительность алгоритма, однако с ростом процента P блоков производительность алгоритма все же незначительно падает (рис. 7).

Из приведенного графика следует, что фактор числа различных меток приписываемых вершинам графов (т. е. числа различных вычислительных узлов, применяемых при формировании конечного автомата) меньше всего влияет на производительность алгоритма. Вне зависимости от значения параметра l , время сравнения одной пары графов остается в очень узких пределах от 0,35 до 0,36 секунды машинного времени. Такое малозначительное влияние параметра l объясняется значительным вкладом то-

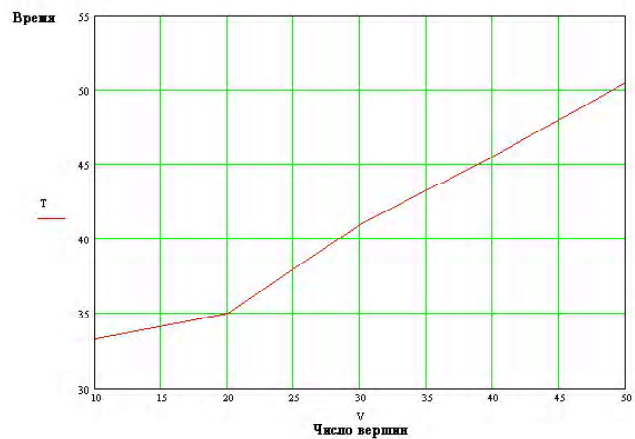


Рис. 6. Зависимость времени поиска подграфа от соотношения P и D блоков в графе конечного автомата (параметры $nv=1000$, $ns=100$, $p=10\dots50$, $l=5$)

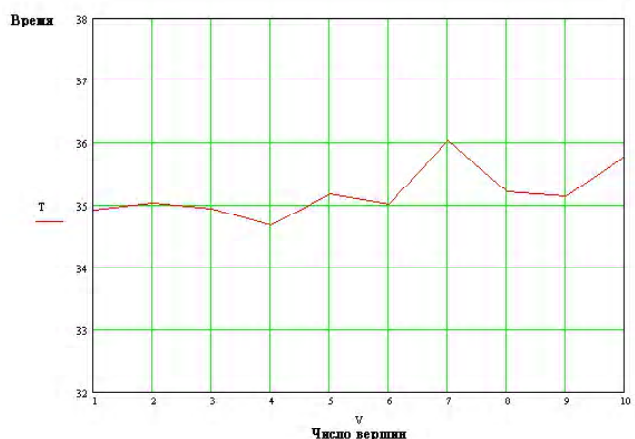


Рис. 7. Зависимость времени поиска подграфа от числа различных меток приписываемых вершинам графов (параметры $nv=1000$, $ns=100$, $p=20$, $l=1..5$)

пологии графов в условие, ограничивающее дерево возможных решений – для случайных графов и графов, не включающих в себя регулярные структуры, большая часть информации используемой для ограничения дерева поиска получается из анализа топологии графов. Но чем больше регулярных структур будут содержать графы, тем более значительным становится влияние фактора меток приписанных вершинам графов.

ВЫВОДЫ

В работе представлено решение задачи поиска граф-подграф изоморфизма для помеченных графов. Приводится детальное описание разработанного алгоритма и результатов исследования его производительности для графов, характерных для конечных автоматов с программируемой процедурой. Применение описанного алгоритма позволяет расширить класс решаемых задач на автоматы с программируемой структурой, что является актуальным [6, 7, 8].

При проектировании проблемно-ориентированных вычислителей на ПЛИС, данное решение обеспечивает оптимизацию аппаратных затрат на этапах ввода задания на проектирование, конфигурирования микропроцессорных систем с программируемой архитектурой, реализации технологий логического, концептуального, функционального программирования и др.

Исследовано влияние различных параметров ГСА на производительность алгоритма поиска подграфа. Результаты исследования показали, что разработанный алгоритм может быть эффективно применен в современных САПР, например, для создания функционально-ориентированных конечных автоматов, полученных в результате анализа концептуальной области проблемно-ориентированным пролог-процессором, одним из способов реализации которого является абстрактная машина Уоррена.

СПИСОК ЛІТЕРАТУРИ

1. *Голдобин, А. А.* Квазигомоморфное преобразование гиперграфов в автоматизации проектирования устройств управления / А. А. Голдобин // Радиоэлектроника, информатика, управление. – 2006. – № 1. – С. 41.
2. *Тейз, А.* Логический подход к искусственному интеллекту: от классической логики к логическому программированию : пер. с франц. / Тейз А., Грибомон П., Луи Ж. и др. – М. : Мир, 1990. – 432 с.
3. *Hassan Ait-Kasi Warren's Abstract Machine : a tutorial reconstruction / Hassan Ait-Kasi Warren's.* – MIT Press, 1999. – 144 p.
4. *Ильяшенко, М. Б.* Разработка и исследование параллельного алгоритма проверки граф-подграф изоморфизма / М. Б. Ильяшенко // Радиоэлектроника, информатика, управление. – 2006. – № 1. – С. 63–69.
5. *Пинчук, В. П.* Основанная на волновом разложении система инвариантов для простых графов и алгоритм распознавания изоморфности / В. П. Пинчук. – К., 1995. – Деп. в ГНТБ Украины 10.05.95, N 1002 – Ук95.
6. *Каляев, А. В.* Многопроцессорные системы с программируемой архитектурой / Каляев А. В. – М. : Радио и связь, 1984. – 240 с.
7. *Баркалов, А. А.* Синтез устройств управления на программируемых логических устройствах / Баркалов А. А. – Донецк : РВА ДонНТУ, 2002. – 262 с.
8. *Соловьев, В. В.* Проектирование цифровых систем на основе программируемых логических интегральных схем / Соловьев В. В. – М. : Горячая линия-Телеком, 2001. – 636 с.

Стаття надійшла до редакції 10.10.2011.

Ильяшенко М. Б., Голдобин О. О.

ВИРІШЕННЯ ЗАДАЧІ ПОШУКУ ІЗОМОРФІЗМУ ГРАФІВ ДЛЯ ПРОЕКТУВАННЯ СПЕЦІАЛІЗОВАНИХ ОБЧИСЛЮВАТЕЛІВ

Пропонується вдосконалений алгоритм пошуку ізоморфізму графів та результати дослідження його ефективності. Об'єктом дослідження є множина граф-схем алгоритмів досягнення мети, що була отримана після обходу семантичної мережі, яка задана, за допомогою абстрактної машини Уоррена.

Ключові слова: декларативна логіка, предикат, дерево виводу, пролог, рекурсивний обхід із поверненням, граф-підграф ізоморфізма.

Il'yashenko M. B., Goldobin A. A.

GRAPH-SUBGRAPH ISOMORPHISM PROBLEM SOLVING FOR DESIGNING SPECIAL COMPUTERS

An advanced algorithm for solving graphs isomorphism problem is proposed and experimental results of its efficiency are presented. Object of investigation is set of control flow graphs of solutions achieved, that were received after circumvent of the semantic network by Warren abstract machine.

Key words: declarative logic, predicate, O-Tree, Prolog, recursively returning, graph-subgraph isomorphism.