УДК 004.021

ЭФФЕКТИВНЫЙ МЕТОД ПОСТРОЕНИЯ ОПТИМИЗИРОВАННОГО ДЕРЕВА ГРАММАТИЧЕСКОГО РАЗБОРА ФОРМУЛ ТЕМПОРАЛЬНОЙ ЛОГИКИ ЛИНЕЙНОГО ВРЕМЕНИ

В.Ю. Мельцов, Г.А. Чистяков

Кафедра электронных вычислительных машин,
ФГБОУ ВПО «Вятский государственный университет», г. Киров;
gennadiychistyakov@gmail.com

Представлена членом редколлегии профессором Н.Ц. Гатаповой

Ключевые слова и фразы: алгоритм маскирования; динамическое программирование; исключающая редукция; предиктивный анализатор.

Аннотация: Представлен алгоритм построения оптимизированного дерева грамматического разбора для формул темпоральной логики линейного времени. Предлагаемый метод позволяет минимизировать число входящих в состав формулы нежелательных темпоральных операторов за время, пропорциональное произведению длины выражения на количество допустимых преобразований.

Введение. При решении широкого спектра прикладных задач, и особенно при их формализации, часто требуются средства для учета временных зависимостей. Подходящим способом отражения порядка событий во времени является использование расширений классической логики высказываний. К таким расширениям относится обладающая мощными описательными возможностями темпоральная логика линейного времени (**LTL**).

В дополнение к базовым логическим операторам ($\vee$, $\wedge$, $\neg$) и константам (*true*, *false*) в этой логике введены шесть временных операторов:

– next (X) – выражение Xp истинно в текущем состоянии, если в следующий момент времени p истинно;

– future (F) – выражение Fp истинно в текущем состоянии, если в будущем p станет истинно;

– globally (G) – выражение Gp истинно в текущем состоянии, если во всех будущих состояниях p будет истинно;

– until (U) – выражение pUq истинно в текущем состоянии, если в будущем q станет истинно, а до тех пор истинно будет p;

– week until (W) – выражение pWq истинно в текущем состоянии, если в будущем q станет истинно, а до тех пор истинно будет p, или же p всегда будет истинно;

– release (R) – выражение pRq истинно в текущем состоянии, если в будущем p станет истинно, а до того момента времени включительно истинно будет q, или же q всегда будет истинно.

В темпоральной логике линейного времени будущее рассматривается как последовательность состояний, поэтому формулы LTL – это формулы пути. Специфика LTL такова, что некоторая формула истинна только в том случае, если она истинна на всех возможных путях развития событий. Более подробные сведения о свойствах LTL могут быть найдены в работах [1, 2].

Темпоральная логика линейного времени нашла широкое применение в области спецификации формальных требований к аппаратным и программным системам. Формулы LTL используются для записи условий, выполнимость которых на модели системы нуждается в проверке. Так как специфицируемые требования, как правило, выражаются в виде формул с большой вложенностью операторов, то для их обработки часто бывает необходимо построение дерева грамматического разбора.

Обобщенный алгоритм построения дерева разбора выражения LTL. Грамматика формулы LTL задается с помощью множества нетерминальных символов (*formula*, *summand*, *factor*, *timepredicate*, *upredicate*, *bpredicate*; стартовый символ – *formula*), множества терминальных символов (логические операции – $\vee$, $\wedge$, $\neg$; временные операции – X, F, G, U, W, R; атомарные предикаты; константы *true* и *false*) и следующих форм Бэкуса–Наура**:**

$$\begin{gathered}
< formula >::=< summand > \{\vee < formula >\};\\
< summand >::=< factor > \{\wedge < formula >\};\\
< factor >::= \{\neg\} < timepredicate >;\\
< timepredicate >::=< upredicate > | < bpredicate > | < apredicate >;\\
< upredicate >::= \{X \mid F \mid G\} < formula >;\\
< bpredicate >::= \{U \mid W \mid R\} < formula >, < formula > .
\end{gathered} \tag{1}$$

Заметим, что в контексте решаемой задачи константы *true* и *false* могут быть приравнены к атомарным предикатам.

Для разбора произвольной формулы темпоральной логики линейного времени и построения дерева грамматического разбора применим основанный на технике предиктивного анализа алгоритм, включающий следующие шаги [3].

1. Разбить заданное выражение на лексемы. Сопоставить итоговому дереву разбора пару вершин. Второй вершине сопоставить операцию «ИЛИ» и отметить ее как текущую.

2. Выбрать очередную лексему. Если такой лексемы не существует, то перейти к пункту 9. Если выбранная лексема – терминальный символ, соответствующий логической операции «ИЛИ», то добавить к текущей вершине дерева разбора новую вершину, сопоставить ей операцию «ИЛИ», отметить добавленную вершину как текущую и повторить пункт 2. Иначе добавить к текущей вершине дерева разбора новую вершину, сопоставить ей операцию «И» и отметить добавленную вершину как текущую. Перейти к пункту 3.

3. Если выбранная лексема – терминальный символ, соответствующий логической операции «И», то запомнить текущую вершину дерева разбора, добавить к текущей вершине дерева две новых вершины, сопоставить им операции «И» и «ИЛИ», отметить последнюю добавленную вершину как текущую и перейти к пункту 2. Иначе перейти к пункту 4.

4. Если выбранная лексема – терминальный символ, соответствующий логической операции «НЕ», то запомнить текущую вершину дерева разбора, добавить к текущей вершине дерева новую вершину, сопоставить ей операцию «НЕ» и отметить добавленную вершину как текущую. Перейти к пункту 5 алгоритма.

5. Если выбранная лексема – терминальный символ, соответствующий унарному модальному оператору, то запомнить текущую вершину дерева разбора, добавить к текущей вершине дерева две новых вершины, сопоставить им операции выбранного модального оператора и «ИЛИ», отметить последнюю добавленную вершину как текущую и перейти к пункту 2. Иначе перейти к пункту 6.

6. Если выбранная лексема – терминальный символ, соответствующий бинарному модальному оператору, то запомнить текущую вершину дерева разбора, добавить к текущей вершине дерева три новых вершины, сопоставить им операцию выбранного модального оператора и логические операции «ИЛИ», отметить вторую добавленную вершину как текущую и повторить пункт 2 для каждой подформулы. Иначе перейти к пункту 7.

7. Если выбранная лексема – атомарный предикат, то добавить к дереву разбора новую вершину, соответствующую выбранному нетерминальному символу. Перейти к пункту 8.

8. В зависимости от промежуточной структуры дерева разбора восстановить текущую вершину (если это необходимо) и вернуться к выполнению соответствующего пункта алгоритма.

9. Алгоритм завершен.

Хотя словесное описание предлагаемого алгоритма кажется сложным, его программная реализация выглядит чрезвычайно компактной и простой. Такой результат достигается благодаря применению рекурсии: все операции сохранения и восстановления текущей вершины дерева разбора выполняются с использованием программного стека. В общих чертах идея предлагаемого метода схожа с идеей метода нисходящего анализа, рассматриваемой в [3]. Нетрудно заметить, что пункты 2–7 алгоритма соответствуют процедурам обработки нетерминальных символов *formula*, *summand*, *factor*, *timepredicate*, *upredicate*, *bpredicate*, а очередная выбранная лексема однозначно определяет поток управления в теле текущей подпрограммы.

Обработка каждой лексемы формулы приводит к добавлению константного числа вершин к дереву разбора $O(1)$, а значит, предлагаемый алгоритм обладает линейной асимптотической сложностью.

Применение описанного алгоритма позволяет построить бинарное дерево грамматического разбора для заданного выражения LTL. Однако в некоторых случаях такое дерево будет содержать избыточные элементы (часть вершин, соответствующих операциям «ИЛИ» и «И», будет иметь только по одной вершине-потомку). Для удаления излишних элементов предлагается использовать следующий алгоритм исключающей редукции.

1. Зафиксировать вершину – корень дерева.

2. Если у зафиксированной вершины есть необработанные вершины-потомки, то поочередно зафиксировать каждого потомка и повторить для него шаг 2. Иначе перейти к шагу 3.

3. Если зафиксированная вершина – корень дерева, то перейти к шагу 4. Иначе, если зафиксированная вершина соответствует операции «ИЛИ» или «И» и имеет только одного потомка – удалить вершину из дерева, присоединив единственного потомка к предку данной вершины. Вернуться к выполнению шага 2 для родительской вершины.

4. Алгоритм завершен.

Алгоритм представляет собой не что иное, как обход дерева в глубину с обработкой вершин на обратном проходе. Такой метод позволяет удалить избыточные вершины путем однократного прохода по дереву. Как было показано выше, количество элементов в дереве растет пропорционально $O(n)$, где n – число лексем обрабатываемой формулы. Следовательно, асимптотическая сложность предлагаемого подхода может быть оценена как $O(n)$.

Комбинация приведенных выше алгоритмов позволяет построить бинарное дерево грамматического разбора для произвольного выражения LTL, содержащее ровно $n+1$ вершину, где n – число лексем в выражении.

Алгоритм оптимизации дерева разбора выражения LTL. В случаях, когда формула LTL используется в дальнейшей обработке данных, важное значение может иметь вид дерева разбора. Например, для перехода от формулы LTL к автомату Бюхи требуется приведение выражения к нормальной форме, которая допускает использование только темпоральных операторов X, U, R.

Темпоральная логика линейного времени разрешает некоторые преобразования над временными операторами. Примерами таких трансформаций могут служить тождества:

$$\begin{array}{lll} G\varphi = \neg F\neg\varphi; & \varphi U\psi = \neg(\neg\varphi R\neg\psi); & F\varphi = trueU\varphi; \\ G\varphi = \neg F\neg\varphi; & \varphi W\psi = (\varphi U\psi)\vee G\varphi; & \varphi W\psi = \psi R(\psi\vee\varphi); \\ \varphi R\psi = \neg(\neg\varphi U\neg\psi); & \varphi R\psi = \neg(\neg\varphi U\neg\psi); & G\varphi = \varphi\wedge X\varphi\wedge XX\varphi\wedge\ldots \end{array} \quad (2)$$

Таким образом, для каждой формулы LTL существует набор эквивалентных формул, а дерево разбора данной формулы имеет набор эквивалентных деревьев грамматического разбора. Для попарного сравнения и оценки структуры таких деревьев необходимо ввести меру, определяющую степень «приспособленности» дерева для дальнейшего использования.

Пусть вектору временных операторов $M = (X, F, G, U, W, R)$ сопоставлен такой вектор $X = (x_1, x_2, x_3, x_4, x_5, x_6)$, что каждый элемент X определяет степень «нежелательности» присутствия каждого соответствующего временного оператора в дереве разбора и $\forall i,\ i = \overline{1,6},\ x_i \in [0,1]$. Тогда для произвольного дерева разбора T и вектора X может быть введена функция $MP(T, X)$, определяющая меру «пригодности» T для дальнейшего использования.

Конкретный вид функции $MP(T, X)$ зависит от специфики решаемой задачи. Например, в ряде ситуаций $MP(T, X)$ может вычисляться:

– как максимальный штраф за вхождение нежелательного временного оператора;

– сумма (или произведение) штрафов за каждое вхождение нежелательного временного оператора в состав дерева разбора выражения;

– полиномиальная функция, понижающая величину штрафов за временные операторы, расположенные на нижних уровнях дерева.

Применение к дереву T подобных преобразований позволяет изменять его структуру, а, значит, и значение целевой функции $MP(T, X)$. В общем случае, минимизация $MP(T, X)$ при заданном наборе правил P может быть выполнена с помощью следующего алгоритма.

1. Зафиксировать вершину – корень дерева.

2. Если у зафиксированной вершины есть необработанные вершины-потомки, то поочередно зафиксировать каждого потомка и повторить для него шаг 2. Иначе перейти к шагу 3.

3. Если зафиксированная вершина – корень дерева, то перейти к шагу 4; если лист дерева, то сопоставить ей нулевое значение целевой функции и вернуться к выполнению пункта 2; если логический оператор, то пересчитать значение целевой функции в соответствии с введенной метрикой и результатами вершин-потомков, вернуться к выполнению пункта 2; если временной оператор, то выбрать из множества преобразований P правило или цепочку правил и заменить зафиксированную вершину, минимизировав целевую функцию. Запомнить значение целевой функции, применить выбранные трансформации для текущей вершины и вернуться к выполнению шага 2 для родительской вершины.

4. Алгоритм завершен.

В основе предлагаемого алгоритма лежит метод динамического программирования – при минимизации функции $MP(T, X)$ для поддерева используются значения аналогичной функции для вершин-потомков.

Под применением цепочки правил следует понимать преобразования вида $\alpha \to \beta$, $\beta \to \gamma$, когда выбранная трансформация приводит к возможности использования другой трансформации. Очевидно, что использование правил, порождающих уже полученные ранее временные операторы, приводит только к увеличению значения целевой функции, и, следовательно, не имеет смысла. Таким образом, длина цепочки правил всегда не превышает пяти элементов (по числу временных операторов).

Необходимо также учесть, что использование цепочек правил может привести к появлению в дереве разбора двойных отрицаний. Такие вершины могут быть удалены на этапе применения к вершине выбранных трансформаций.

Пусть $m = |P|$, а n – число элементов в дереве грамматического разбора. На шаге 3 предложенного алгоритма выполняется построение цепочки трансформаций для вершины дерева, отмеченной темпоральным оператором, с целью минимизации значения функции $MP(T, X)$ поддерева. С учетом вышесказанного шаг 3 может быть выполнен за время, пропорциональное $O(m)$. Для выбора очередного правила цепочки используется стандартный алгоритм маскирования: каждый бит шестиразрядной маски указывает на возможность применения правил, порождающих соответствующий временной оператор. Исчерпывающие сведения об особенностях реализации данного подхода приведены в [5]. Так как общее число вершин дерева разбора, соответствующих временным операторам, не более n, то итоговая временная сложность рассматриваемого алгоритма не превосходит $O(nm)$.

Заметим, что несмотря на свою экспоненциальную сложность, метод маскирования не ухудшает асимптотическую оценку, так как число видов возможных временных операторов всегда равно шести.

Алгоритм построения оптимизированного дерева грамматического разбора формулы LTL. Для произвольного выражения LTL Ф и набора допустимых темпоральных преобразований P оптимизированное дерево грамматического разбора может быть построено по следующему алгоритму.

1. Построить дерево грамматического разбора для выражения Ф.

2. С помощью алгоритма исключающей редукции удалить из полученного дерева избыточные вершины.

3. Определить вектор штрафов X и функцию $MP(T, X)$.

4. Минимизировать значение функции $MP(T, X)$, применяя к дереву T преобразования из множества P. Алгоритм завершен.

Асимптотическая оценка временных затрат предлагаемого алгоритма определяется наиболее трудоемким этапом минимизации значения целевой функции и составляет $O(nm)$.

Пример. Пусть оптимизированное дерево грамматического разбора требуется построить для выражения

$$\Phi = \neg(\varphi \vee ((\neg\psi W\varphi) \wedge F\varphi).$$

Множество допустимых преобразований P включает следующие правила:

$$F\varphi = \neg G\neg\varphi; \qquad G\varphi = falseR\varphi;$$

$$\psi W\varphi = \psi U(\varphi \vee G\psi); \qquad \varphi R\psi = \psi W(\psi \wedge \varphi); \tag{3}$$

$$F\varphi = \varphi \vee X\varphi \vee XX\varphi \vee XXX\varphi.$$

Последняя формула означает, что в контексте решаемой задачи формулировка «когда-нибудь в будущем» может быть заменена на более сильную «в течение трех следующих временных квантов» (известно, что событие всегда происходит в ходе трех ближайших моментов времени или не происходит вовсе).

Дерево разбора, полученное в результате выполнения первого шага алгоритма, представлено на рис. 1.

Выполнение второго шага алгоритма приводит к получению дерева разбора, не содержащего избыточных вершин. Редуцированное дерево изображено на рис. 2.

Пусть функция $MP(T, X)$ вычисляется как сумма штрафов за использование временных операторов, а вектор X равен $(0.05, 0.4, 0.7, 0.1, 1.0, 0.4)$. Обозначим через $Q(T, N)$ операцию выбора в дереве T поддерева с корнем в вершине N. Тогда в результате выполнения пункт 4 алгоритма значения целевой функции для поддеревьев примут следующие значения.

Значения $MP(Q(T,7), X)$, $MP(Q(T,8), X)$, $MP(Q(T,9), X)$, $MP(Q(T,11), X)$ равны 0.0.

Начальное значение $MP(Q(T,6), X)$ составляет 1.0, однако, к вершине 6 может быть применено правило $\psi W\varphi = \psi U(\varphi \vee G\psi)$, которое уменьшит $MP(Q(T,6), X)$ до 0.8. Использование данного правила позволит заменить оператор G оператором R по правилу $G\varphi = falseR\varphi$. Такое преобразование сократит значение $MP(Q(T,6), X)$ до 0.5. Правило $\varphi R\psi = \psi W(\psi \wedge \varphi)$ не сможет быть применено, так как порождение оператора W будет запрещено посредством маскирования.

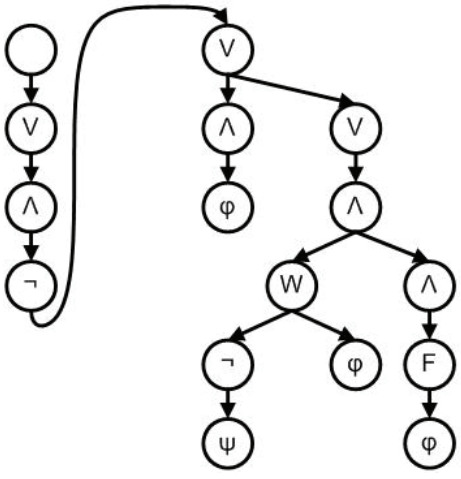

Рис. 1. Дерево разбора выражения Φ

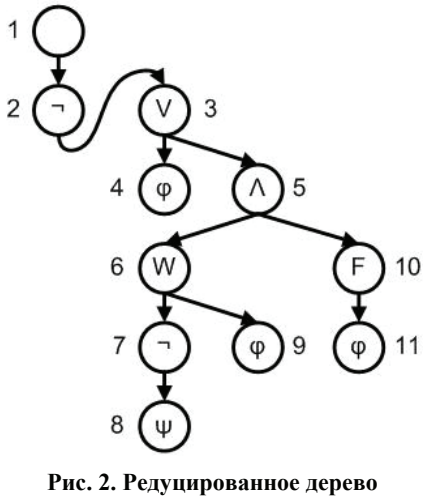

Рис. 2. Редуцированное дерево разбора выражения Φ

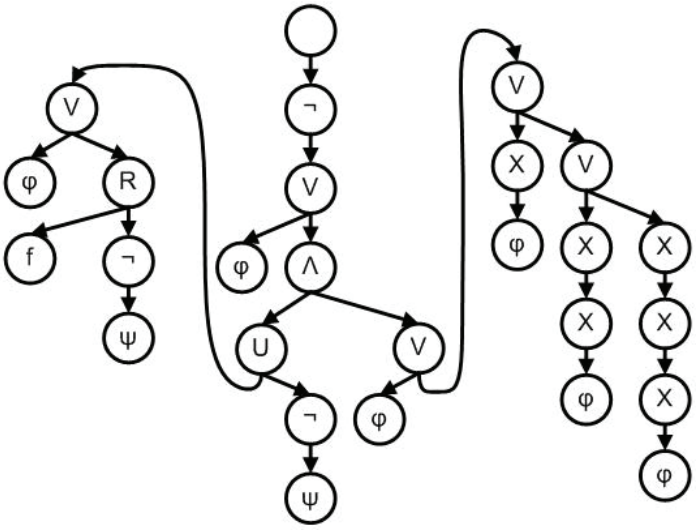

Рис. 3. Оптимизированное дерево разбора выражения Ф

Схожим образом значение $MP(Q(T,10),X)$ будет уменьшено с 0.4 до 0.3 посредством выбора из возможных цепочек преобразований ($F\varphi = \neg G\neg\varphi$; $F\varphi = \varphi \vee X\varphi \vee XX\varphi \vee XXX\varphi$; $F\varphi = \neg G\neg\varphi \rightarrow G\varphi = falseR\varphi$; $F\varphi = \neg G\neg\varphi \rightarrow \rightarrow G\varphi = falseR\varphi \rightarrow \varphi R\psi = \psi W(\psi \wedge \varphi)$) наиболее оптимальной трансформации по правилу $F\varphi = \varphi \vee X\varphi \vee XX\varphi \vee XXX\varphi$.

Полученное дерево разбора представлено на рис. 3.

Таким образом, применение представленного алгоритма позволило снизить значение целевой функции для выражения Ф с 1.4 до 0.8.

Заключение. В работе предложен эффективный метод построения оптимизированного дерева грамматического разбора формул темпоральной логики линейного времени. Данный алгоритм позволяет перестроить дерево разбора в соответствии с допустимым набором преобразований и введенной метрикой. Метод рекомендуется к применению в тех случаях, когда дерево выступает в качестве промежуточной структуры данных для дальнейшей обработки информации.

В настоящее время рассмотренный алгоритм используется в модуле спецификации требований, входящем в состав программного комплекса для верификации параллельных алгоритмов [6].

Список литературы

1. Automata, Languages and Programming / S. Abramsky [and al.]. – Springer-Verlag Berlin Hiedelberg, 2010. – 613 с.

2. Baier, C. Principles of Model Checking / C. Baier, J.-P. Katoen. – Cambridge : The MIT Press, 2008. – 994 с.

3. Компиляторы: принципы, технологии и инструментарий / А. Ахо [и др.]. – М. : Вильямс, 2007. – 1184 с.

4. Московские олимпиады по информатике / под ред. Е.В. Андреевой, В.М. Гуровица, В.А. Матюхина. – М. : Изд-во МЦНМО, 2006. – 256 с.

5. Василевский, Б. Динамическое программирование по профилю [Электронный ресурс] / Б. Василевский. – Режим доступа : http://ejudge.btty.su/bmstu/2007-2008/docs/dp2.pdf. – Загл. с экрана (дата обращения: 30.10.2012).

6. Meltsov, V.Yu. Development Modules for Specification of Requirements for a System of Verification of Parallel Algorithms / Vasiliy Yu. Meltsov, Gennadiy A. Chistyakov // European Researcher. – 2012. – Vol. (20), № 5-1. – P. 511–514.

Effective Method for Constructing Optimized Parse Tree of Temporal Logic Formulas of Linear Time

V.Yu. Meltsov, G.A. Chistyakov

Department of Electronic Computing Machines, Vyatka State University, Kirov; gennadiychistyakov@gmail.com

Key words and phrases: dynamic programming; masking algorithm; predictive analyzer; reduction with eliminating.

Abstract: This paper presents an algorithm for constructing optimized parse tree for the formulas of temporal logic of linear time. The proposed method minimizes the number of unwanted temporal operators in the formula with time complexity proportional to the product of the expression length and the number of admissible transformations.

Wirksame Methode des Aufbaues des optimisierten Baumes der grammatikalischen Untersuchung der Formelen der Temporallogik der linearen Zeit

Zusammenfassung: Es ist den Algorithmus der Konstruktion des optimisierten Baumes der grammatikalischen Untersuchung für die Formelen der Temporallogik der linearen Zeit dargelegt. Die angebotene Methode lässt zu, die Zahl der in die Bestandteile der Formel ungewünschten eingehenden Temporaloperatoren für die Zeit, die zu das Produkt der Länge des Ausdruckes auf die Zahl der zulässigen Umgestaltungen proportional ist, zu minimisieren.

Méthode efficace de la construction de l'arbre optimisée de l'analyse grammaticale des formules de la logique temporaire du temps linéaire

Résumé: Est présenté un algorithme de la construction de l'arbre optimisée de l'analyse grammaticale des formules de la logique temporaire du temps linéaire. La méthode proposée permet de minimiser le nombre des opérateurs temporaires indésirables entrant dans la formule en temps qui est proportionnel au produit de la longeur de la formule sur la quantité des transformations admissibles.

Авторы: ***Мельцов Василий Юрьевич*** – кандидат технических наук, доцент кафедры электронных вычислительных машин; ***Чистяков Геннадий Андреевич*** – аспирант кафедры электронных вычислительных машин, ФГБОУ ВПО «Вятский государственный университет», г. Киров.

Рецензент: ***Иномистов Валентин Юрьевич*** – кандидат технических наук, доцент, заведующий кафедрой прикладной математики и информатики, ФГБОУ ВПО «Вятский государственный университет», г. Киров.