УДК 004.832.3

# ПРИМЕНЕНИЕ ПОЗИТИВНО-ОБРАЗОВАННЫХ ФОРМУЛ ДЛЯ ИССЛЕДОВАНИЯ ЧАСТИЧНО-НАБЛЮДАЕМЫХ ДСС

**А.В. Давыдов**

*Институт динамики систем и теории управления им. В.М. Матросова СО РАН*

Россия, 664033, Иркутск, Лермонтова ул., 134

E-mail: artem@icc.ru

**А.А. Ларионов**

*Институт динамики систем и теории управления им. В.М. Матросова СО РАН*

Россия, 664033, Иркутск, Лермонтова ул., 134

E-mail: bootfrost@zoho.com

**Н.В. Нагул**

*Институт динамики систем и теории управления им. В.М. Матросова СО РАН*

Россия, 664033, Иркутск, Лермонтова ул., 134

E-mail: sapling@icc.ru

**Ключевые слова:** позитивно-образованная формула; автоматическое доказательство теорем; дискретно-событийная система; супервизорное управление; частичная наблюдаемость.

**Аннотация:** В докладе показана связь теории управления частично наблюдаемыми дискретно-событийными системами и автоматическим доказательством теорем (АДТ) в исчислении позитивно-образованных формул (ПОФ). Язык ПОФ – это полноценный язык первого порядка, предоставляющий мощный инструмент качественного анализа динамических систем с использованием АДТ. На основе ПОФ предлагается новая методика для проверки наблюдаемости и реализации супервизорного управления ДСС. При нарушении наблюдаемости показано, как с помощью ПОФ извлекаются слова, вызывающие конфликт.

## 1. Введение

Доклад посвящен расширению применения исчисления позитивно-образованных формул (ПОФ) и автоматического доказательства теорем (АДТ) в задачах теории супервизорного управления (ТСУ) дискретно-событийными системами (ДСС). Современное состояние ТСУ представлено в [1], [2]. Рассматривается случай частично-наблюдаемых ДСС, когда возникновение некоторых событий в системе недоступно для наблюдения. В этом случае свойство *наблюдаемости* формального языка определяет те ограничения на функционирование ДСС,

которые могут быть обеспечены супервизорным управлением. На сегодняшний день известно несколько эффективных алгоритмов проверки наблюдаемости, например, [3] и [4], где рассматриваются функция наблюдения в форме маски и алгебраические операции над процессами, представляющими ДСС, соответственно. Для регулярных языков можно эффективно реализовать проверку наблюдаемости на основе метода фиксированной точки из [5]. В докладе будет показано, как полиномиальный алгоритм из [1] может быть реализован с помощью ПОФ и АДТ, а также как с помощью ПОФ могут быть найдены конфликтующие строки. Основным преимуществом представленного подхода на основе ПОФ и использования метода АДТ является декларативное описание используемых алгоритмов. В этом случае программист лишь описывает (декларирует) свойства искомого результата, а решение (метод) предоставляется системой логического программирования в результате поиска логического вывода некоторого целевого утверждения. Это шаг вперед по сравнению с программированием на императивных языках (например, C/C++, Java и т. д.), поскольку программисту не нужно беспокоиться о низкоуровневых деталях программы.

## 2. Исчисление ПОФ

Исчисление позитивно-образованных формул (ПОФ) основано на опровержении отрицания исходного утверждения, истинность которого необходимо установить. Язык ПОФ – это язык логики первого порядка (ЛПП), который состоит из первопорядковых логических формул (ПЛФ), построенных из атомарных формул, или атомов, с помощью связок & , $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, кванторных символов $\forall$ и $\exists$, а также констант $True$ и $False$. Любая формула ЛПП может быть представлен в виде ПОФ, поскольку ПОФ-язык представляет собой особый способ записи классических ПЛФ (ср. нормальные формы ЛПП). Алгоритм преобразования представлен в [6].

Для облегчения чтения ПОФ можно представлять графически в виде древовидной структуры. Корень $\forall\varnothing$ ПОФ-дерева называется *корнем* ПОФ. Дочерний узел $\exists_X A$ каждого корня ПОФ называется *базой* ПОФ, конъюнкт $A$ называется *базой фактов*, а максимальная подформула ПОФ, корнем которой является база, называется *базовой подформулой*. Узлы-потомки $\forall_Y B$ базовой подформулы называются *вопросами* для соответствующей базы. Поддеревья (т.е. подформулы) вопросов называются *консеквентами*. Если вопрос не имеет консеквент, то этот вопрос называется *целевым вопросом* а сам консеквент равносилен $False$.

Единственная аксиома ПОФ-исчисления – $\forall\varnothing\colon \varnothing$, т. е. $False$. Правило вывода $\omega$ в ПОФ-исчислении основано на поиске так называемых *ответных подстановок*, т.е. таких замен переменных в термах, которые удовлетворяют определенному условию. Любая конечная последовательность ПОФ $\mathcal{F}, \omega\mathcal{F}, \omega^2\mathcal{F}, \ldots, \omega^n\mathcal{F}$, где $\omega^s\mathcal{F} = \omega(\omega^{s-1}\mathcal{F}), \omega^1 = \omega, \omega^n\mathcal{F} = \forall$, называется *выводом* $\mathcal{F}$ в исчислении ПОФ (с аксиомой $\forall$). Ответ на целевой вопрос опровергает соответствующую базу. Когда все базы ПОФ опровергнуты, то ПОФ $\mathcal{F}$ сводится к $\forall$, т.е. $False$. Это означает, что $\mathcal{F}$ как отрицание рассматриваемого утверждения невыполнимо; следовательно, само утверждение истинно. Подробности о ПОФ-исчислении можно найти в [7–9].

$$\mathcal{F}_{Obs} = \exists B_{Obs} \begin{cases} R_1 \\ R_2 \\ \dots \\ R_6 \end{cases}$$

$$R_1 : \forall q_G, q_H^1\ Q_0^G(q_G), Q_0^H(q_H^1) - \exists\ Q_0^T(q_H^1, q_H^1, q_G), T(q_H^1, q_H^1, q_G)$$

$$R_2 :\ \forall \sigma, q_H^1, q_H^2, q_G\ T(t_H^1, q_H^2, q_G), E_c(\sigma), F_H(q_H^1, \sigma), NoF_H(q_H^2, \sigma), F_G(q_G, \sigma) - \exists\ dead(q_H^1, q_H^2, q_G, \sigma)$$

$$R_3 : \forall q_H^1, q_H^2, q_G, \sigma\ dead(q_H^1, q_H^2, q_G, \sigma)$$

$$R_4 : \forall \sigma, q_H^1, q_H^2, q_G, t_H^1, t_H^2, t_G\ T(q_H^1, q_H^2, q_G), E_o(\sigma), \delta_H(q_H^1, \sigma, t_H^1), \delta_H(q_H^2, \sigma, t_H^2), \delta_G(q_G, \sigma, t_G) - \exists\ T(t_H^1, t_H^2, t_G), \delta_T(q_H^1, q_H^2, q_G, \sigma, \sigma, \sigma, t_H^1, t_H^2, t_G)$$

$$R_5 : \forall \sigma, q_H^1, q_H^2, q_G, t_H^1\ T(q_H^1, q_H^2, q_G), E_{uo}(\sigma), \delta_H(q_H^1, \sigma, t_H^1) - \exists\ T(t_H^1, q_H^2, q_G), \delta_T(q_H^1, q_H^2, q_G, \sigma, \epsilon, \epsilon, t_H^1, q_H^2, q_G)$$

$$R_6 : \forall \sigma, q_H^1, q_H^2, q_G, t_H^2, t_G\ T(q_H^1, q_H^2, q_G), E_{uo}(\sigma), \delta_H(q_H^2, \sigma, t_H^2), \delta_G(q_G, \sigma, t_G) - \exists\ T(q_H^1, t_H^2, t_G), \delta_T(q_H^1, q_H^2, q_G, \epsilon, \sigma, \sigma, q_H^1, t_H^2, t_G)$$

Рис. 1. ПОФ для проверки наблюдаемости.

## 3. Проверка наблюдаемости спецификации

Предположим, что частично-наблюдаемая ДСС задана конечным автоматом $G$, и $L(G)$ – генерируемый ею язык. Пусть спецификация на $G$ задана регулярным языком $K$, и $H$ – распознающий его конечный автомат. Рассмотрим алгоритм проверки наблюдаемости $K$ из [1], полиномиальный относительно размерности автомата $H$. Основная идея алгоритма заключается в построении автомата, который отслеживает два слова $s_1$, $s_2$ из $K$, имеющих одну и ту же проекцию $P(s_1) = P(s_2)$, но такие, что $s_1 \in \overline{K}$, в то время как $s_2 \notin \overline{K}$. Строки $s_1$ и $s_2$ называются конфликтными, поскольку они требуют разных управляющих действий со стороны супервизора, а содержащая их спецификация является неуправляемой. Алгоритм из [1] предлагает рассмотреть два экземпляра автомата $H$ и один экземпляр автомата $G$ и построить автомат $T$ с состояниями вида $(h_1, h_2, q)$, где $h_1 \in Q_H$, $h_2 \in Q_H$, $q \in Q_G$, и единственным состоянием $dead$, существование которого означает ненаблюдаемость $K$, поскольку в этом случае для $s_i \in \overline{K}$, $i = 1, 2, 3$ и некоторого события $\sigma$ такого, что $s_2 = s_3$ и $P(s_1) = P(s_2)$, $s_1\sigma \in \overline{K}$, $s_3\sigma \in L(G)$ пока $s_2\sigma \notin \overline{K}$. Покажем, как автомат $T$ может быть построен с помощью вывода специальной ПОФ.

Далее будет использоваться следующий список предикатов: предикат $Q_0^X(\_)$, соответствующий начальным состояниям автомата $X$, предикат $E_o(\_)$, который определяет все наблюдаемые события автомата $X$, предикат $E_{uo}(\_)$, который определяет все ненаблюдаемые события автомата $X$, предикаты $\delta_X(q_1^i, \sigma^i, q_2^i)$, определяющие переходы из состояния $q_1^i$ автомата $X$ в состояние $q_2^i$ в результате возникновения события $\sigma^x$. Пусть терм $F_X(q^i, \sigma^j)$ означает, что существует переход

$$\mathcal{F}_{Conf} = \exists B_{Conf} \lessdot \begin{matrix} R_1^{Conf} \\ R_2^{Conf} \end{matrix}$$

$$R_1^{Conf} : \forall \sigma, q_H^1, q_H^2, q_G, q_H^1 f, q_H^2 f, q_G f, s_1, s_2, s_3\ \delta_T(q_H^1 f, q_H^2 f, q_G f, s_1, s_2, s_3, q_H^1, q_H^2, q_G),$$
$$dead(q_H^1, q_H^2, q_G, \sigma) - \exists\ next(q_H^1 f, q_H^2 f, q_G f, s_1, s_2, s_3, \sigma)$$

$$R_2^{Conf} : \forall \sigma, q_H^1, q_H^2, q_G, q_H^1 f, q_H^2 f, q_G f, s_1, s_2, s_3, \sigma_1, \sigma_2, \sigma_3\ next(q_H^1, q_H^2, q_G, s_1, s_2, s_3, \sigma),$$
$$\delta_T(q_H^1 f, q_H^2 f, q_G f, \sigma_1, \sigma_2, \sigma_3, q_H^1, q_H^2, q_G) -$$
$$\exists\ next(q_H^1 f, q_H^2 f, q_G f, \sigma_1 \cdot s_1, \sigma_2 \cdot s_2, \sigma_3 \cdot s_3, \sigma)$$

Рис. 2. ПОФ для извлечения конфликтных строк.

из состояния $q^i$ автомата $X$, отмеченного событием $\sigma^j$. Пусть $NoF_X(q^i, \sigma^j)$ означает обратное, т.е. что такого перехода нет.

Рассмотрим ПОФ $\mathcal{F}_{Obs}$ (рис. 1) с базой $B_{Obs} = B'_{PrepG} \cup B'_{PrepH} \cup \{Q_0^G(q_0^G), Q_0^H(q_0^H), E_c(\sigma^j), E_o(\sigma^j), E_{uo}(\sigma^j)\}$. Вопрос $R_1$ добавляет к базе атом $T(q_H, q_H, q_G)$, который запускает вывод. Второй вопрос проверяет, не нарушает ли контролируемое событие $\sigma$ условие наблюдения, и в случае успешного ответа в базу добавляется атом $dead(q_H^1, q_H^2, q_G, \sigma)$. Содержит информацию о состоянии и событии, при которых были нарушены условия наблюдения. Третий вопрос – целевой вопрос, ответ на который завершает процесс поиска вывода. Остальные вопросы обрабатывают наблюдаемые и ненаблюдаемые события, реализуя шаги алгоритма [1].

Если язык спецификации оказался ненаблюдаемым, ПОФ $\mathcal{F}_{Conf}$ позволяет извлечь из автомата $T_{obs}$ конфликтующие строки (рис. 2). Начальной базой ПОФ $\mathcal{F}_{Conf}$ выбирается база $B'_{Obs}$, полученная в ходе проверки наблюдаемости, $B_{Conf} = B'_{Obs}$.

# 4. Представление системы с помощью ПОФ

Пусть $\Gamma : Q \to 2^\Sigma$ – *функция активных событий*: $\Gamma(q)$ – множество всех события $e$, для которых определено $\delta(q, e)$. $\Gamma(q)$ также называется *набором активных событий* (или набором допустимых событий) $G$ в $q$ [1]. Известно, что замкнутое поведение объекта и супервизора может быть реализовано путем параллельной композиции соответствующих автоматов, т.е. $L(J/G) = L(H||G)$. Для решения BSCOP и реализации супервизорного управления в случае частичного наблюдения событий в $G$ будем использовать автомат-наблюдатель $H_{Obs}$ автомата $H$, порождающего спецификацию. Пусть $s$ – текущая строка ДСС $G$ и $t = P(s)$ – наблюдаемая в данный момент строка. Пусть $x_{obs}$ – текущее состояние $H_{Obs}$ после выполнения $t$. Тогда $J_P(t) = \bigcup_{x \in x_{obs}} [\Gamma_{H_{Obs}}(x)]$ где $\Gamma_{H_{Obs}}$ – функция активного события $H_{Obs}$. Чтобы использовать ту же технику, которая использовалась для случая полного наблюдения событий, достаточно для каждого состояния $x_{obs}$ добавить циклы для тех ненаблюдаемых событий, которые активны в соответствующих $x \in x_{obs}$ в $H$ [1]. Полученный автомат $H'_{Obs}$ гарантирует желаемый результат $L(J_P/G) = L(H'_{Obs}||G)$ для супервизора $J_P$.

Поведение системы под супервизорным управлением может быть описано ПОФ

$$\mathcal{F}_{BSCOP} = \exists\, B_{BSCOP} - \overset{\forall \sigma, s, q, q', \bar{q}_J, \bar{q}'_J\ L^{J_P/G}(s,q), \delta^G(q,\sigma,q'),}{\delta^{H'_{Obs}}(\bar{q}_J, \sigma, \bar{q}'_J), q \in \bar{q}_J} - \exists\, L^{J_P/G}(s \cdot \sigma, q')$$

Рис. 3. ПОФ для управляемой системы.

$\mathcal{F}_{BSCOP}$ (рис. 3). Здесь предикат $L^{J_P/G}(\_, \_)$ используется для хранения слов управляемого языка $L^{J_P/G}$ в качестве первого аргумента. Состояние, в которое эта строка привела систему, сохраняется как второй аргумент. Хотя язык $L(J_P/G)$ является ограниченной версией $L(G)$, мы используем новый предикат $L^{J_P/G}(\_, \_)$, чтобы подчеркнуть, что построенный язык является результатом совместной работы объекта управления и супервизора. База $B_{BSCOP}$ состоит из наборов атомов, соответствующих переходам объекта и супервизора соответственно, и содержит исходный атом $L^{J_P/G}(\varepsilon, 1)$.

# 5. Заключение

В дальнейших работах представленный подход к проверке наблюдаемости будет расширен для решения других задач ТСУ, в том числе для построения наибольших наблюдаемых подъязыков ненаблюдаемых спецификаций. Для реализации представленного и планируемых подходов используется прувер Bootfrost, разработанный для поиска логических выводов на языке ПОФ [10].

# Список литературы


1. Cassandras C.G., Lafortune S. Introduction to Discrete Event Systems. Cham: Springer, 2021.
2. Wonham W.M., Kai C. Supervisory control of discrete-event systems. 2019.
3. Cho H., Marcus S.I. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations // Mathematical systems theory. 1989. Vol. 22. P. 177–211.
4. Inan, K. An algebraic approach to supervisory control // Math. Control. Signals Syst. 1992. Vol. 5. P. 151–164.
5. Rudie K., Wonham W.M. The infimal prefix-closed and observable superlanguange of a given language // Systems & Control Letters. 1990. Vol. 15. P. 361–371.
6. Давыдов А.В., Ларионов А.А., Черкашин Е.А. Метод трансляции первопорядковых логических формул в позитивно-образованные формулы // Программные продукты и системы. 2019. Т. 32, № 4. С. 197–206.
7. Vassilyev S.N. Machine Synthesis of Mathematical Theorems // The Journal of Logic programming. 1990. Vol. 9, No 2-3. P. 235-266.
8. Васильев С.Н., Жерлов А.К., Федунов Е.А., Федосов Б.Е. Интеллектное управление динамическими системами. 2000. М.: Физико-математическая литература, 352 с.
9. Davydov A.V., Larionov A.A., Cherkashin E.A. On the calculus of positively constructed formulas for automated theorem proving // Automatic Control and Computer Sciences (AC&CS). 2011. Vol. 45, No. 7. P. 402-407.
10. Larionov A. Bootfrost, https://github.com/snigavik/bootfrost (дата обращения: 18.01.2024).