УДК 681.3

Дорошенко А, Ю., Алістратов О. В.

ЗАСОБИ ЛОГІЧНИХ СПЕЦИФІКАЦІЙ РЕАКТИВНИХ АЛГОРИТМІВ

Двійкові діаграми рішень відомі як ефективний засіб подання систем булевих функцій і широко застосовуються в проектуванні цифрових систем, аналізі скінченних автоматів, у задачах штучного інтелекту та математичної логіки. У цій роботі розглянуто питання представлення логічних специфікацій реактивних алгоритмів та запропоновано представлення у вигляді двійкових діаграм рішень як таке, що передбачає ефективну реалізацію алгоритмів синтезу та верифікації реактивних систем.

Вступ

Останнім часом значного поширення набули формальні методи синтезу і верифікації реактивних алгоритмів [1], специфікованих у вигляді сукупності формул однієї з логічних мов. Це можуть бути мови логіки 1-го і 2-го порядку з одномісними предикатами [2, 3], темпоральні логіки [4], числення на основі операторів найменшої і найбільшої нерухомої точки (μ-числення) та ін. Використання формальних методів проектування або верифікації алгоритму передбачає наявність специфікації даного алгоритму в деякій формальній мові.

Водночас проблеми верифікації, синтезу та оптимізації цифрових систем потребують детальної характеристики реактивного алгоритму або його частин у вигляді скінченного автомата. Класичні алгоритми для таких задач [4] використовують явне представлення графа станів. Однак такі методи не є практичними, тому що кількість станів сягає значних величин (наприклад, один 32-бітний регістр може мати понад 4×109 станів), а розмір представлення лінійно пропорційний кількості станів. Тому для задач синтезу та верифікації цифрових систем, зокрема реактивних (тобто таких, що вимагають узгодженості своєї поведінки із середовищем перебування), потрібні методи, що забезпечать задовільну за часом обчислення та за обсягом використаної пам'яті реалізацію.

Останнім часом дедалі більше дослідників у пошуках зручної та ефективної структури даних звертаються до двійкових діаграм рішень (Binary Decision Diagrams, BDD). Двійкові діаграми рішень представляють систему булевих функцій у вигляді направленого ациклічного графа. Операції над булевими функціями можуть бути реалізовані як графові алгоритми над структурами даних BDD.

У цій статті розглянуто питання ефективності застосування BDD як засобу подання логічної специфікації та автоматної моделі реактивного алгоритму порівняно з іншими методами на конкретному прикладі. Як приклад розглянемо одну з найпростіших реактивних систем: механізм взаємовиключного доступу, так званий мутекс (Mutual Exclusion Locks, mutex). Мутекси застосовуються для організації доступу до розподіленого ресурсу (наприклад, фізичного пристрою або ділянки пам'яті), який може бути використаний одночасно тільки одним процесом, для того, щоб гарантувати, що критична секція коду (частина коду, де відбувається звернення до розподіленого ресурсу) виконується в кожен момент часу тільки одним процесом. Практично мутекс є різновидом семафора, який сигналізує іншим процесам, що критична секція коду вже виконується.

Таким чином, складемо неформальний опис функціонування мутекса, який керує доступом двох процесів до певного ресурсу. Нехай процес 1 посилає вхідний сигнал X_1 у той момент часу, коли він потребує доступу до ресурсу, і протягом усього наступного часу до того моменту, коли потреба в ресурсі відпадає. Відповідно, процес 2 використовує для цієї мети вхідний сигнал X_2. Проектована система повинна видати вихідний сигнал Y_1, якщо доступ дозволено першому процесу, та Y_2 – другому процесу. В довільний момент часу може бути виданий сигнал Y_1 або Y_2, але не обидва разом (власне властивість взаємовиключення). Нарешті, система повинна надавати незайнятий ресурс безпосередньо в момент приходу вхідного сигналу.

Засоби формальних специфікацій реактивних алгоритмів

У цій роботі для формальної специфікації використовується мова L, яка є фрагментом ло-

гіки предикатів 1-го порядку з одномісними предикатами та фіксованою областю інтерпретації – множини Z цілих чисел з відношенням лінійного порядку. Докладний опис цієї мови можна знайти в [5].

Алфавіт мови складається з таких символів:

1) символ предметної константи 0, що інтерпретується як ціле число 0;
2) множина предметних змінних Var = $\{t, t_1, t_2, \ldots\}$, яка визначається конкретною специфікацією;
3) сигнатура одномісних предикатних символів $\Omega = \{p_1, \ldots, p_q\}$, що також визначається конкретною специфікацією;
4) двомісний предикатний символ $\leq$, який інтерпретується як відношення лінійного порядку на множині цілих чисел;
5) два унарних функціональних символи +1 і –1, що інтерпретуються як додавання та віднімання одиниці;
6) логічні зв'язки &, $\vee$, $\neg$, квантори $\forall$, $\exists$ та дужки (,).

Множина T термів складається з двох підмножин $T_0 = Z$ (константні терми) та $T_1 = \{(t_i + k) \mid t_i \in \text{Var}, k \in Z\}$, які не перетинаються. Атомарні формули мови мають вигляд $p_i(\tau_j)$ і $\tau_i \leq \tau_j$ ($p_i \in \Omega$, $\tau_i \in T_1$, $\tau_j \in T$). Інші формули мови будуються з атомарних формул за допомогою логічних зв'язок і кванторів. На вид формул, які використовуються для специфікації функціональних властивостей алгоритму, накладаються певні обмеження [6].

Детально процес побудови формальної специфікації за певними неформальними функціональними вимогами до функціонування проектованого алгоритму описано в [7]. Тому формальну специфікацію даної системи механізму взаємного виключення подамо вже в остаточному вигляді.

Вихідний сигнал має значення 1 тільки в тому випадку, якщо значення відповідного вхідного сигналу дорівнює 1:

$$Y_1(t) \to X_1(t); \quad Y_2(t) \to X_2(t).$$

Вихідний сигнал, встановлений в 1, зберігає це значення, доки значення відповідного вхідного сигналу дорівнює 1:

$$Y_1(t-1) \,\&\, X_1(t) \to Y_1(t);$$
$$Y_2(t-1) \,\&\, X_2(t) \to Y_2(t).$$

У будь-який момент лише один вихідний сигнал дорівнює одиниці:

$$\neg Y_1(t) \vee \neg Y_2(t);$$

На зміну вхідного сигналу система відповідає негайно (в той самий проміжок дискретного часу):

$$X_1(t) \to (Y_1(t) \vee Y_2(t)); \; X_2(t) \to (Y_1(t) \vee Y_2(t)).$$

Таким чином, формальна специфікація реактивного алгоритму є логічною формулою (вказаною під квантором узагальнення $\forall t$, який для спрощення випускається). Для алгоритмів синтезу автоматної моделі доцільно розглядати таку формулу в кон'юнктивній нормальній формі (КНФ). Дана специфікація після перетворення в КНФ, поповнення та певних специфічних для алгоритмів синтезу [5] спрощень матиме такий вигляд:

$$\neg Y_1(t) \vee X_1(t) \,\&\, \neg Y_2(t) \vee X_2(t) \,\&$$
$$\&\; Y_1(t) \vee \neg X_1(t) \vee \neg Y_1(t-1) \,\&$$
$$\&\; Y_2(t) \vee \neg X_2(t) \vee \neg Y_2(t-1) \,\&\, \neg Y_1(t) \vee \neg Y_2(t) \,\&$$
$$\&\; Y_1(t) \vee Y_2(t) \vee \neg X_1(t) \,\&\, Y_1(t) \vee Y_2(t) \vee \neg X_2(t) \,\&$$
$$\&\; Y_2(t) \vee X_1(t) \vee \neg X_2(t) \,\&\, Y_1(t) \vee \neg X_1(t) \vee X_2(t) \,\&$$
$$\&\; \neg Y_2(t) \vee \neg X_1(t) \vee \neg Y_1(t-1) \,\&$$
$$\&\; \neg Y_1(t) \vee \neg X_2(t) \vee \neg Y_2(t-1).$$

Отже, мова L є початковим засобом представлення формальної специфікації (подібно до мов програмування високого рівня, вона передбачає створення специфікації людиною). Для зручності машинної обробки логічна функція переводиться в канонічну форму. Часто на практиці застосовують представлення множини диз'юнктів або кон'юнктів у табличному вигляді, де для кожної змінної вказується, чи входить вона до диз'юнкту (кон'юнкту), і якщо входить, то з яким «знаком», тобто для нашого прикладу:

Таблиця 1. **Логічна специфікація в КНФ, представлена у вигляді бітових полів**

№	$X_1(t)$	$X_2(t)$	$Y_1(t)$	$Y_2(t)$	$X_1(t-1)$	$X_2(t-1)$	$Y_1(t-1)$	$Y_2(t-1)$
1	+		–					
2		+		–				
3	–		+				–	
4		–		+				–
5			–	–				
6	–		+	+				
7		–	+	+				
8	+	–		+				
9	–	+	+					
10	–			–			–	
11		–	–					–

Подібні структури даних потребують значного обсягу пам'яті, тому зазвичай вони реалізуються у вигляді упакованих бітових полів (для утримання стану змінної достатньо двох бітів). Подібне подання логічної функції є очевидним, але має такі недоліки:

1. Деякі функції породжують надто великі представлення.

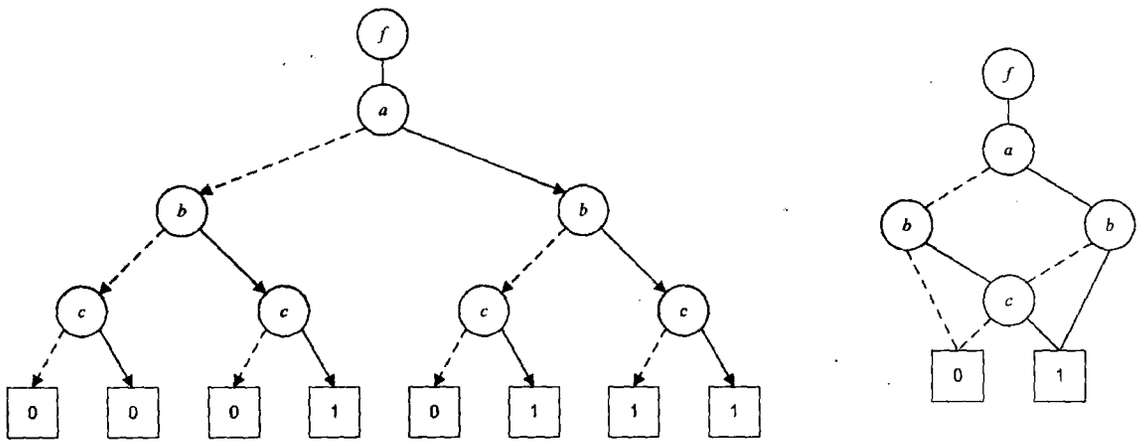

Рис. 1. Дерево рішень та редукована впорядкована діаграма двійкових рішень для функції $F = a \,\&\, b \vee a \,\&\, c \vee \neg a \,\&\, b \,\&\, c$

2. Одержання КНФ з ДНФ та навпаки потребує значних обчислювальних витрат (поповнення, кон'юнкція формул в ДНФ, диз'юнкція формул в КНФ є досить складними).
3. Перевірка еквівалентності, істинності КНФ і тавтологічності ДНФ також є затратними в плані ефективності задачами.

Усі ці недоліки успішно уникаються при застосуванні двійкових діаграм рішень – BDD-представлень. Хоча BDD відомі відносно давно [8], лише представлені Р. Брайянтом (Randal Bryant) у 1986 р. ефективні алгоритми маніпуляції BDD [9] зробили їх популярним інструментом у галузі проектування та верифікації цифрових систем і привернули увагу дослідників. Брайянт визначив, що редуковані та впорядковані BDD (Reduced Ordered BDD, ROBDD) є канонічним представленням логічних булевих формул. Надалі, говорячи про двійкові діаграми рішень, ми матимемо на увазі саме редуковані впорядковані BDD.

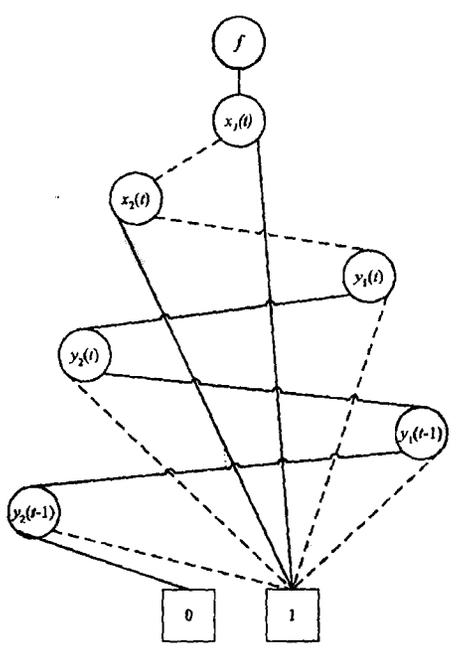

Рис. 2. Упорядкована редукована двійкова діаграма даних, що відповідає вихідній специфікації

Схематично дерево рішень для булевої функції має такий вигляд (рис. 1).

Вершина f дерева називається *вузлом функції*. Проміжні вершини, позначені іменами змінних, назвемо *внутрішніми* вузлами, а вузли, позначені 1 (істина) та 0 (хибність), назвемо *кінцевими* вузлами. Обчислення значення функції F полягає в простеженні шляху від вузла функції до кінцевого вузла, позначення якого буде шуканим значенням функції. З кожного внутрішнього вузла шлях іде за суцільною дугою, якщо відповідна вузлу змінна має істинне значення, інакше – за штриховою дугою. Порядок, в якому перелічуються змінні, однаковий за всіма шляхами (перша за порядком змінна є найближчою до вузла функції).

У результаті процесу *редукції* з дерева рішень отримуємо відповідну двійкову діаграму рішень. Процес редукції полягає в застосуванні двох правил до початкового дерева рішень доти, доки ці правила можуть бути застосовані:

1. Якщо два вузли є кінцевими і мають однакове позначення або є внутрішніми і мають однакових нащадків, вони об'єднуються (правило об'єднання).
2. Якщо внутрішній вузол має ідентичних нащадків, він вилучається з графа і його вхідні вузли перенапрямляються до нащадка (правило видалення).

На практиці дерево рішень не будується і згодом редукується. Замість того BDD створюються з констант і змінних застосуванням звичайних логічних зв'язок і постійно підтримуються в редукованому вигляді. Крім того, кілька функцій можуть бути представлені однією діаграмою з декількома вузлами функцій.

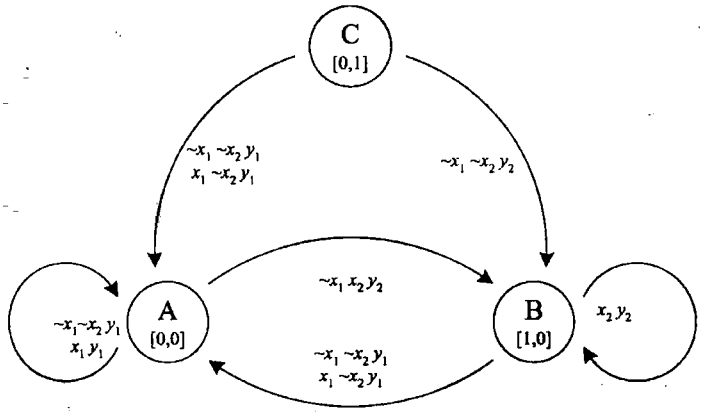

Рис. 3. Явне представлення скінченного автомата, побудованого за вихідною специфікацією

Таким чином, двійковою діаграмою рішень називається напрямлений ациклічний граф, що представляє вектор булевих функцій. Повне визначення BDD, як і умови, за яких вони є канонічною формою булевих функцій, можна знайти, наприклад, у [10].

Усі $_{дії}$ над BDD можуть бути здійснені алгоритмами, які мають поліноміальну складність, залежно від кількості змінних та функцій [11, 12]. У результаті символічні булеві маніпуляції, базовані на BDD, мають дві переваги над іншими підходами. По-перше, доки граф залишається прийнятного розміру, повне його обчислення залишається задовільним за складністю. Наочним прикладом є одержання ДНФ логічної формули, яке полягає просто у виборі всіх шляхів на графі, що ведуть від функціонального вузла до кінцевого вузла 1. По-друге, хоча розмір графа може зростати з кожною наступною дією, будь-яка окрема дія навіть у найгіршому випадку має прийнятну складність. Більшість інших представлень логічних функцій не має такої властивості. Наприклад, поповнення множини диз'юнктів, представлених у вигляді бітових полів, може мати експоненціальну складність.

BDD-представлення скінченного автомата

Більшість логічних мов специфікації має автоматну семантику, тобто формулам мови ставиться у відповідність скінченний автомат (або клас еквівалентних скінченних автоматів). Таким чином, алгоритми синтезу реактивних систем породжують, як правило, скінченний автомат у тому чи іншому вигляді; ця математична модель передбачає, наприклад, подальше застосування відомих алгоритмів мінімізації автоматів чи побудову опису алгоритму в довільній алгоритмічній мові.

Класичним випадком є зображення скінченного автомата у вигляді направленого графа, вершини якого відповідають станам автомата, а дуги – переходам з цих станів до інших при заданих сигналах.

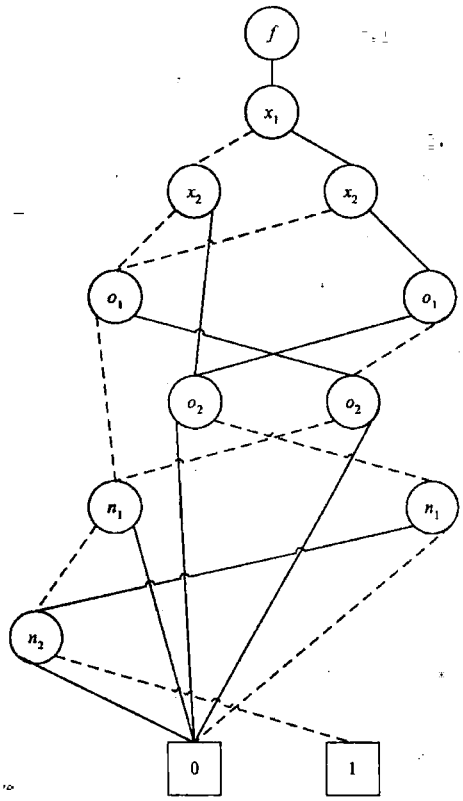

Рис. 4. Символічне представлення скінченного автомата, побудованого за вихідною специфікацією

Як уже було зазначено, розмір такого представлення лінійно пропорційний кількості станів. Тому частіше використовується символічне представлення скінченних автоматів, в якому структурі переходу між станами відповідає булева функція. Кількість змінних у такому випадку залежить від кількості станів логарифмічно. Окремим випадком такого представлення і є двійкові діаграми рішень.

Для подання скінченного автомата у вигляді BDD слід спочатку вибрати двійкове кодування станів системи та вхідного алфавіту. Поведінка наступного стану описується характеристичною функцією $\delta(\bar{x},\bar{o},\bar{n})$, яка дорівнює 1, якщо вхідний вектор $\bar{x}$ викликає перехід від стану $\bar{o}$ до стану $\bar{n}$. На рис. 3 зображено граф станів детермінованого скінченного неініціального автомата, який був синтезований за специфікацією мутекса. Він має BDD-представлення, зображене на рис. 4. Для позначення трьох можливих станів використовується два двійкових значення (δ(A) = [0, 0], δ(B) = [1, 0], δ(C) = [0, 1]), змінні x_1, x_2 відповідають вхідним сигналам. Невикористане значення [1, 1] може трактуватися як фіктивне; у даному випадку воно використовується як альтернативне значення для стану C, щоб спростити BDD-представлення.

Для такого маленького автомата представлення у вигляді BDD не дає переваг над явним

представленням, але для складніших систем BDD може бути значно меншого розміру. За деяких умов BDD-представлення відношень переходу зростає лінійно з ростом числа компонентів автомата, тоді як число станів зростає експоненційно.

Висновок

У статті описано спосіб подання логічної специфікації та автоматної моделі реактивних алгоритмів засобами двійкових діаграм розв'язань (BDD) та показано ефективність його використання для задач синтезу і верифікації. Запропонований спосіб проілюстровано на прикладі. BDD є універсальним представленням логічної інформації протягом усього життєвого циклу проектування реактивних систем. Переваги використання BDD ставлять перспективну задачу розробити таку реалізацію алгоритмів синтезу та верифікації реактивних систем, яка б застосовувала BDD для всіх необхідних маніпуляцій.

1. *Harel D., Pnueli A.* On the Development of Reactive Systems // in K. R. Apt, ed., Logic and Models of Concurrent Systems.– NATO ASI Series, vol. F13.– Springer, Berlin, 1985.– P. 477–498.
2. *Henriksen J. G., Jensen J.* et al. Mona: Monadic Second Order Logic in Practice // First International Workshop, TACAS'95, LNCS.– 1995.– 1019.– P. 89–110.
3. *Чеботарев А. Н.* Расширение логического языка спецификации и проблема синтеза // Кибернетика и системный анализ.– 1996.– № 6.– С. 11–27.
4. *Clarke E. M., Emerson E. A., Sistla A. P.* Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specification // ACM Transactions on Programming Languages, April 1986.– P. 244–263.
5. *Чеботарев А. Н.* Синтез процедурного представления автомата, специфицированного в логическом языке L* // Кибернетика и системный анализ.– 1997.– № 4.– С. 60–74.
6. *Чеботарев А. Н.* Об одном способе спецификации реактивных алгоритмов в логическом языке первого порядка // Проблемы программирования.– 2000.– № 1–2.– С. 273–279.
7. *Чеботарев А. Н., Алистратов О. В.* Построение логической спецификации реактивного алгоритма // Проблемы программирования.– 2002.– № 1.– С. 154–160.
8. *Akers S. B.* Binary Decision Diagrams // IEEE Transactions on Computers, C-27(6), June 1978.– P. 509–516.
9. *Bryant R. E.* Graph-Based Algorithms for Boolean Function Manipulation // IEEE Transactions on Computers, C-35(8), August 1986.– P. 677–691.
10. *Somenzi F.* Binary Decision Diagrams. http://www.cs.chalmers.se/ComputingScience/Education/Courses/svh/Papers/CSD-BDDs.ps.gz, p. 5–6.
11. *Bryant R. E.* Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams // ACM Computing Surveys, Vol. 24, № 3, September 1992.– P. 293–318.
12. *Mishchenko A. A.* An Efficient Implementation of L Language Data Processing Algorithms // Проблемы программирования.– 2000.– № 1–2.– С. 335–344.

A. Yu. Doroshenko, O. V. Alistratov

THE MEANS OF LOGIC SPECIFICATIONS OF REACTIVE ALGORITHMS

Binary decisions diagrams are known as effective means of representation of systems Boolean functions, and they are widely applied in digital systems design, finite-state system analysis, problems of artificial intelligence, and mathematical logic. Issues of representations of logic specifications of reactive algorithms are considered in this paper, and binary decision diagrams are offered as such which provides efficient realization of synthesis and verification algorithms for reactive systems.