УДК 004.94(075.8)

Н.В. Богушевська, М.А. Баляниця

АВТОМАТИЗАЦІЯ СИНТЕЗУ ПРОГРАМНОГО КОДУ ВУЗЛА МЕРЕЖІ ПЕТРІ ДЛЯ МОВИ МОДЕЛЮВАННЯ POSES++

Аннотация: Проведений огляд систем моделювання мережами Петрі, показані переваги використання в якості зовнішнього середовища моделювання CASE-пакету системи POSES++. Визначена формальна модель вузла, задана контекстно-вільна граматика для синтезу програмного коду вузлу. Розглянуто приклад застосування правил граматики.

Ключові слова: мережі Петрі, CASE-пакет, POSES++, правила граматики.

Вступ

Теорія мереж Петрі є добре відомим засобом формального опису, основне призначення якого – моделювання та аналіз роботи з паралельними та асінхроними системами. У мережі Петрі вузол характеризує деякий стан системи, а перехід – дію, що відбувається у системі. Система, перебуваючи в певному стані, може породжувати дії; виконання дій переводить систему з одного стану в інший.

Наразі існує багато систем, які дозволяють проводити моделювання з використанням мереж Петрі. Проте більшість з систем являють собою деякий спеціалізований інструментарій, для вільного володіння яким необхідно бути не лише фахівцем із мереж Петрі, а й фахівцем відповідного програмного продукту.

Альтеративним шляхом є використання так званих CASE- засобів, які дозволяють користувачеві будувати модель за допомогою візуального інтерфейсу та здійснювати параметричне налаштування побудованої моделі, після чого побудова програмного коду здійснюється за деякою формалізованою схемою [1]. У роботі [2] розглянуті особливості взаємодії CASE- засобу із системою імітаційного моделювання, та обґрунтована доцільність використання системи імітаційного моделювання як окремого застосування. При цьому на вхід застосування подається файл з програмним кодом імітаційної моделі, а на виході отримується файл із результатами проведення експерименту над моделлю.

Проте цей підхід можливий лише у тому випадку, коли система моделювання має відкритий формат зберігання файлів із програмним кодом та результатами експериментів. Характеристика систем моделювання мережами Петрі, згідно із даними [3], наведена у наступній таблиці:

Для інтеграції із CASE-засобом можуть бути використані системи моделювання Maria, PNML FrameWork, POSES++ та SNAKES, так як вони мають відкритий формат збереження моделей та не мають власного графічного інтерфейсу.

 ISSN 1560-8956

Таблиця 1

Характеристика систем моделювання мережами Петрі

Назва системи	Має відкритий формат зберігання даних	Має візуальний інтерфейс	Переваги	Недоліки
CPN Tools	+	+	Зручний графічний інтерфейс	Формат збереження моделі базується на мові XML. Вбудовані засоби аналізу моделі недостатні для вирішення прикладних задач
Geist3D	-	-	Тривимірне представлення моделі, запис відео-результатів моделювання	Відсутні можливості анімації результатів
Maria	+	-	Представлення моделі здійснено з використанням формальної мови з можливістю розширення	Обмежені можливості аналізу моделі
Petrigen	-	+	Використання мови декларативної мови для представлення моделі	
PNML Frame Work	+	-	Сумісність з багатьма інструментами візуального представлення моделей засобами мереж Петрі	

Найпотужнішою з перелічених систем моделювання є система POSES++, яка може застосовуватися для моделювання дискретно-подійних систем (логістика, апаратне забезпечення) та реалізована у вигляді клієнт-серверного застосування з можливістю розподілених обчислень.

Роботи [4,5] присвячені питанням розробки формальних моделей

Назва системи	Має відкритий формат зберігання даних	Має візуальний інтерфейс	Переваги	Недоліки
POSES++	+	-	Можливість реалізації розподілених обчислень.	Вимагає знання мови С
SNAKES	+	-	Широкі функціональні можливості створення та редагування моделі	Вимагає знання мови Python
Snoopy	-	+	Розвинена система візуального представлення процесу симуляції моделі	Не підтримує кольорові мережі Петрі
StpnPlay	+	+	Використання засобів багато потокового моделювання	Відсутні можливості анімації результатів
Tina	-	+	Потужні аналітичні можливості, можливість збереження результатів у різних форматах	Не підтримує кольорові мережі Петрі
TAPAAL	+	+	Потужні аналітичні можливості аналізу та верифікації моделей	
Visual Object Net ++	-	+	Можливість створення та аналізу гібридних моделей (дискретно-неперервних). Генерація звітності по проведених експериментах	

основних конструкцій мови POSES++ та правил перетворення формальної моделі типу даних до програмного коду мовою POSES++.

Постановка задачі

Для розробки правил перетворення вузла мережі Петрі до програмного коду мовою POSES++ необхідно:

 ISSN 1560-8956

1. визначити алфавіт термінальних символів вузлу A_{nodes};
2. визначити алфавіт нетермінальних символів вузлу N_{nodes};
3. визначити початковий символ S_{nodes};
4. визначити множину правил виводу R_{nodes};
5. створити діалогову нотацію відповідно до формальної моделі.

Формальна модель вузла

Визначення алфавіту термінальних символів

Формальна модель вузлу мережі Петрі має такий вигляд [1]:

$$\text{Node} =< \text{Name, TokenType, Capacity, AccessMode, Initial} > , \quad (1)$$

де Name – найменування вузла, TokenType $\in$ Token – тип даних маркера у вузлі, Capacity – ємність вузла, AccessMode $\in$ Access – режим доступу у вузлі, Initial – множина значень, що визначають початкову розмітку вузлу.

Множина Access визначає можливі режими доступу маркерів у вузлі, для мови POSES++ :

$$\text{Access} = \{ \text{RAM, FIFO, LIFO, FIFORAM, LIFORAM,PLACE} \}. \quad (2)$$

Алфавіт A_{nodes}представимо у вигляді

$$A_{nodes} = A^0_{nodes} \cup A'_{nodes}, \quad (3)$$

де A^0_{nodes}– містить термінали, що відповідають операторам мови POSES++, A'_{nodes} – містить термінали “найменування”.

Визначимо множину можливих роздільників A_{del} та множину типів даних A_{types} наступним чином:

$$A_{\text{types}} = \{\text{'char', 'short', 'int', 'long', 'float', 'double', '\$', 'string', 'enum', 'struct'}\}, \quad (4)$$

$$A_{del} = \{\text{';', ',', '}', '\{', '<', '>', '(', ')'}\}. \quad (5)$$

Тоді

$$A^0_{nodes} = A_{types} \cup A_{del} \setminus \{\text{'enum','struct','\{','}','(',')'}\}. \quad (6)$$

Позначимо через A_{acc} алфавіт режимів доступу:

$$A_{acc} = \{\text{'ram', 'fifo', 'lifo', 'fiforam', 'liforam', 'place'}\}, \quad (7)$$

Визначення алфавіту нетермінальних символів та початкового символу

Множини алфавіту не термінальних символів N_{nodes} та початкового символу S_{nodes} матимуть вигляд:

$$N_{nodes} = \{Node, Node1, Node2, AccessMode1, AccessMode2, TokenType, LPar, RPar, Coma, Number, Name, Variable, Initial, End\}, \quad (8)$$

$$S_{nodes} = \{\text{ Node }\}. \quad (9)$$

Визначення множини правил виводу

Множина правил виводу R_{nodes} :

1. < Node >::=< Node1 > | < Node2 >
2. < Node1 >::=< AccessMode1 >< LPar >< TokenType >< Coma > < Number >< RPar >< Name >< Initial >
3. < Node2 >::=< AccessMode2 >< LPar >< Number >< RPar >< Name >< Initial >
4. < AccessMode1 >::= ram | fifo | lifo | fiforam | liforam <
5. < AccessMode2 >::= place
6. < TokenType >::= char | short | int | long | float | double | string | < Variable >
7. < LPar >::=<
8. < RPar >::=>
9. < Coma >::= ,
10. < Name >::= Name
11. < Number >::= Number
12. < Variable >::= Variable
13. < Initial >::=< LPar >< LPar > Number (< Initial >)| < End > | card(Number) < Coma >< BlackToken >
14. < End >::= ,
15. < BlackToken >::= $

Алгоритм застосування правил 1) – 15) для отримання програмного коду має такий вигляд:

1. Визначення типу вузла. Якщо вузол містить чорні маркери, то застосувати правило 3, інакше – правило 2.

2. Поки є нетермінальні символи, застосовувати найменше з правил 4) – 15).

 ISSN 1560-8956

Приклад застосування

Розглянемо приклад. Нехай формальна модель (1) має вигляд

$$\text{Node} = < \text{n1, int, 5, lifo,\{1,2,3\}} > .$$

Для цього прикладу $A^{=}_{nodes}$\{n1,5,\{1,2,3\}\}. Покрокове застосування правил R_{nodes}, з урахуванням (4) – (9), матиме наступний вигляд:

Правило 1

$$< \text{Node} > :: = < \text{Node1} >$$

Правило 2

$$< \text{Node1} > :: = < \text{AccessMode1} >< \text{LPar} >< \text{TokenType} >< \text{Coma} > < \text{Number} >< \text{RPar} >< \text{Name} >< \text{Initial} >$$

Правила 7 та 8

$$< \text{Node1} > :: = < \text{AccessMode1} >< \text{TokenType} >< \text{Coma} >< \text{Number} >> < \text{Name} >< \text{Initial} >$$

Правило 4

$$< \text{Node1} > :: = \text{lifo} << \text{TokenType} >< \text{Coma} >< \text{Number} >> < \text{Name} >< \text{Initial} >$$

Правило 6

$$< \text{Node1} > :: = \text{lifo} < \text{int} < \text{Coma} > < \text{Number} >>< \text{Name} >< \text{Initial} >$$

Правило 9

$$< \text{Node1} > :: = \text{lifo} < \text{int} , < \text{Number} >>< \text{Name} >< \text{Initial} >$$

Правило 10

$$< \text{Node1} > :: = \text{lifo} < \text{int} , < \text{Number} >> \text{n1} < \text{Initial} >$$

Правило 11

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} < \text{Initial} >$$

Правило 13

$$< \text{Node1} >::= \text{lifo} < \text{int} ,5 > \text{n1} < \text{LPar} >< \text{LPar} > 1 < \text{Initial} >$$

Правило 7

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} << 1 < \text{Initial} >$$

Правило 13

$$< \text{Node1} >::= \text{lifo} < \text{int} ,5 > \text{n1} << 1 < \text{LPar} >< \text{LPar} > 2 < \text{Initial} >$$

Правило 7

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} << 1 << 2 < \text{Initial} >$$

Правило 13

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} << 1 << 2 < \text{LPar} > < \text{LPar} > 3 < \text{Initial} >$$

Правило 7

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} << 1 << 2 << 3 < \text{Initial} >$$

Правило 13

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} << 1 << 2 << 3 < \text{End} >$$

Правило 14

$$< \text{Node1} > :: = \text{lifo} < \text{int} ,5 > \text{n1} << 1 << 2 << 3 ,$$

Висновки

Запропоновані контекстно-вільно граматика та формальні моделі дають змогу автоматизувати синтез програмного коду вузла мережі Петрі. Побудова множини правил граматики здійснена у відповідності до синтаксису мови POSES++, тому покрокове застосування правил дозволяє отримати синтаксично та семантично вірний код мовою POSES++.

Література

1. Томашевский В.Н., Купрашвили А.Ю., Савустьяненко Э.И. Организация интерактивной системы моделирования // Электронное моделирование. – 1987. – 1. – С. 16–19.
2. Богушевська Н.В., Звіряка Є.Д. Взаємодія CASE-пакету ISS2005 із системою імітаційного моделювання //III Всеукраїнська наук.-техн. конф. молодих учених та студентів "Інформаційні технології в економічних та технічних системах –2009", м.Кременчук – 2009. – С.145 –147.
3. Complete Overview of Petri Nets Tools Database / Режим доступу http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools
4. Баляница Н.А., Богушевская Н.В. Определение формальных моделей основных моделируемых конструкций языка POSES ++ для расширения возможностей системы ISS 2000// Міжн. наук. конф. "Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій", м. Євпаторія. – 2009. – т.1. с.12–15.
5. Баляница Н.А., Богушевская Н.В. Определение правил создания типов данных языка POSES++ для использования в системе ISS 2000// Четверта науково-практична конференція з міжнародною участю "Математичне та імітаційне моделювання систем МОДС '2009", м. Київ. С.184–188.

Отримано 11.03.2011 р.

 ISSN 1560-8956