Декларативен и императивен подходи за доказване на Тюринг пълнота на SPIDER/CNP

Цанко Големанов

Abstract: ***Control Network Programming (CNP)*** *is a style of high-level programming that is especially effective for solving problems that have natural graph-like representation of imperative, declarative, or mixed nature. A* ***Turing machine (TM)*** *is a theoretical abstraction that expresses the extent of the computational power of algorithms. Any system that is Turing complete is sufficiently powerful to recognize all possible algorithms. The report is aimed to provide a very simple proof of Turing-completeness of SPIDER/CNP by declarative and imperative coding recursive functions thereby showing that Turing-completeness is a consequence of a few basic and fundamental features of the SPIDER-language[1].*

Key words: *Turing-completeness, Control Network Programming, CNP, SpiderCNP.*

ВЪВЕДЕНИЕ

Машина на Тюринг е абстрактно изчислително устройство, описано от английския математик Алън Тюринг [2], с което той определя и границите на ефективната изчислимост – това е устройство, което може да се справи с кое да е изчисление за достатъчно дълъг, но краен срок.

Доказването на Тюринг пълнота е сред основните изисквания, поставяни пред всеки изчислителен модел, като за целта е достатъчно той да може да симулира работата на стандартна Тюринг машина.

Една Тюринг машина, включва следните основни елементи:

- Безкрайна лента (Памет), състояща се от клетки, във всяка от които има по един символ;
- Глава за чете/запис на един символ от текущата клетка на лентата;
- Блок вътрешни състоянията на машината;
- Логически блок (Управляващо устройство).

Действието на машината на Тюринг протича на тактове. При всеки такт се изпълнява една команда, при което управляващото устройство на базата на прочетения символ и текущото вътрешно състояние, определя какъв символ да се запише на лентата, кое да бъде следващото вътрешно състояние и в коя посока да се премести главата за четене/запис.

Формално Тюринг-машина M може да се дефинира [3] чрез наредената седморка:

$$M = (Q, \Sigma, \Gamma, \delta, \mathbf{q}_0, \square, F)$$

където:

Q	е крайно множество от вътрешни състояния
Σ	е крайно множество от символи, наречени входна азбука
Γ	е крайно множество от символи, наречено азбука на лентата
δ	е функция на преходите
$\mathbf{q}_0 \in Q$	е начално състояние
$\square \in \Gamma$	е специален „празен“ символ
$F \subseteq Q$	е множество финални състояния

Входната азбука се дефинира като подмножество на азбуката на лентата, невключващо „празния“ символ:

$$\Sigma \subseteq \Gamma - \{\square\}$$

Функцията на преходите δ е частична функция и се дефинира като:

$$\delta : \quad Q \times \Sigma \rightarrow Q \times \Gamma \times \{L,R\}$$

Нейни аргументи са текущото вътрешно състояние и текущо прочетения символ от лентата. В резултат управляващото устройство генерира новото вътрешно състояние, новия символ, който трябва да се запише на мястото на прочетения и знак за посока L или R. Означението за посока определя накъде да се придвижи главата за четене/запис след извършване операцията върху лентата.

Работата на Тюринг машината преминава през последователност от стъпки, контролирани от управляващото устройство. Процесът продължава до изпадане в halt-състояние, което може да се случи или при конфигурация, когато функцията δ е недефинирана (функцията е частична), или при достигане на финално състояние, от което на практика също няма дефинирани преходи.

Тюринг машината може да работи или като разпознавател (проверява за коректност началното съдържание на лентата) или като преобразувател (модифицира съдържанието на лентата според правилата от функцията δ).

По-долу ще бъдат разгледани два подхода за симулиране на Тюринг машина посредством средствата, с които CNP/SPIDER [1][4] разполага.

Декларативен подход за симулиране на стандартна Тюринг машина

Това е по-естествения подход, от гледна точка на факта, че ориентацията на CNP е към декларативната програмна парадигма. Основния акцент на този подход е декларирането на вътрешните състояния и функцията на преходите δ директно като насочен граф (CNP-подмрежа).

CNP-моделирането на Тюринг машина при декларативния подход се базира на следните правила:

1. Декларират се като глобални променливи с подходящ формат лентата на Тюринг машината (tape) и позицията на главата за четене/запис (head);
2. Създава се управляваща мрежа (УМ), състояща се от две подмрежи;
3. Главната подмрежа е с едно нормално състояние и една стрелка, насочена от него към състояние FINISH;
4. По стрелката към FINISH се поставят:
   a. Примитив за инициализиране на глобалните променливи на Тюринг машината (начално съдържание на лентата и позиция на главата за четене/запис);
   b. Системен примитив CALL за обръщение към подмрежа (ориентиран граф) задаваща функционалността на управляващото устройство;
5. Създава се подмрежа за графа на управляващото устройство, като на всяко едно от множество от вътрешни състояния Q се съпоставя състояние от подмрежата;
6. Добавя се състояние RETURN, към което се насочват стрелки от всички финални състояния F;
7. Състоянията на подмрежата се свързват със стрелки с примитиви отразяващи функционалността на δ, като за правило от вида:

$$\delta\,(q_i, s_r) = (q_j, s_w, D)$$

се създава стрелка насочена от състояние q_i към състояние q_j.

8. По всяка стрелката се поставят двойка примитиви, симулиращи необходимите проверки и модификации:

a. Примитив-Условие Test(s_r) – пропуска напред само ако текущо-прочетения символ на лентата съвпада с s_r;
b. Примитив-Действие WriteAndMove(s_w,D) – записва в клетката на лентата символа s_w и премества главата една позиция според знака за посока D ∈ {L,R}.

Тестов пример: Нека е дадена Тюринг машина [3], дефинирана чрез:

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \square\}$
$F = \{q_1\}$

и функция на преходите:

$\delta(q_0, a) = (q_0, b, R)$
$\delta(q_0, b) = (q_0, b, R)$
$\delta(q_0, \square) = (q_1, \square, L)$

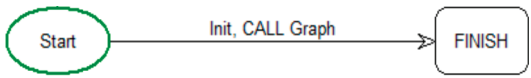
В този случай, CNP-моделирането на Тюринг машина се базира на следната последователност:

1. Декларират се като глобални променливи лентата на Тюринг машината (tape) и позицията на главата за четене/запис (head)

```
Var   tape   : string;
      head   : integer;
```

2. Създава се УМ, състояща се от две подмрежи: MAIN StandartTuringMachine и SUB Graph

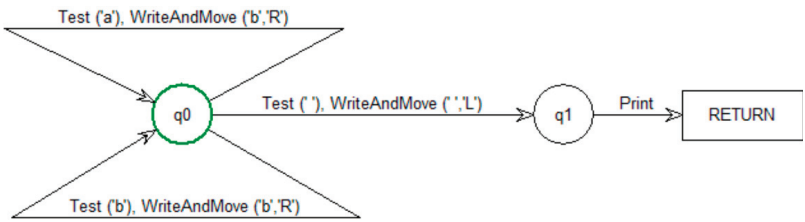
MAIN StandartTuringMachine



Предназначението на главната подмрежа е да инициализира Тюринг машината чрез примитива Init (начално съдържание на лентата и позиция на главата за четене/запис) след което се извършва обръщение към подмрежа Graph.

3. В подмрежа Graph е дефиниран графа на конкретната Тюринг машина, описващ вътрешните й състояния (q_0, q_1) и преходите между тях, задавани от функцията δ.

SUB Graph



4. По стрелките на УМ се разполагат примитиви, отразяващи логиката на δ:

Например за $\delta(q_0, a) = (q_0, b, R)$ се създава стрелка насочена от q_0 към q_0, по която се преминава само ако текущия символ на лентата е a. Ако това условие е

изпълнено, то в тази клетка на лентата се записва b, а главата се премества една позиция надясно (R):

(q_0) → Test ('a'), WriteAndMove ('b','R') → (q_0)

5. Примитивите се дефинират в SpiderUnit

```
procedure Test (read : char);
 begin
 if FORW then
    FAILURE:= tape[head]<>read;          // Пропуска напред само, ако текущия символ на лентата
                                         // съвпада с параметъра read. В противен случай се индицира
                                         // локален неуспех (FAILURE) и управлението се връща назад.
 end;

 procedure WriteAndMove (write, direction : char);
 begin
 if FORW then
    begin
      tape[head]:=write;                          // Запис на параметъра write в текущата позиция
                                                  // на лентата
      if direction = 'R'     then inc(head)       // Преместване на главата (head) в посока,
                             else dec(head);      // зададена от параметъра direction: R:+1, L:-1
    end
 end;
```

Вижда се, че примитивите са елементарни и използват съвсем базово множество от оператори на host-езика – такива за присвояване и условна конструкция.

Примитивът Print се използва единствено за извеждане на новото съдържание на лентата при достигнато финално (q_1) състояние.

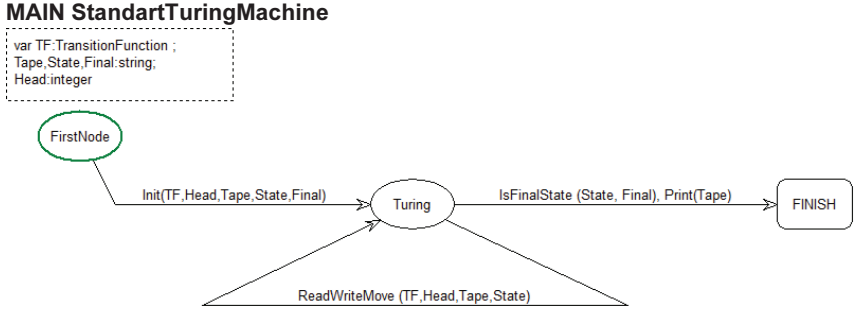
Императивен подход за симулиране на стандартна Тюринг машина

При този подход основната тежест пада върху обработки, задавани глобално в примитивите чрез възможностите на host-езика. Ролята на CNP-програмата е ограничена до организацията на изчислителния процес. Подходът има по-универсален характер (не е свързан с конкретна Тюринг машина), поради факта, че функцията δ се задава като структура и не е вградена в УМ на SPIDER програмата. Това обаче сериозно утежнява обработките свързани с извличането и интерпретацията на правилата, според които функционира Тюринг машината.

Тук CNP-моделирането на Тюринг машина се базира на следната последователност:

1. На глобално ниво се дефинира тип TransitionFunction, задаващ в подходящ формат структурата на правилата от функцията δ
2. Създава се УМ, състояща се от една главна подмрежата StandartTuringMachine
3. Като мрежови променливи се декларират всички необходими структури данни

```
Var
  TF                  : TransitionFunction;   // текуща функция δ
  Tape, State, Final  : string;               // лента, текущо и финално състояния
Head                  : integer;              // позиция на главата за четене/запис
```



4. От състояние FirstNode излиза стрелка с примитив Init за начална инициализация на всички използвани мрежови променливи.
5. От състояние Turing излизат две стрелки:
   a. Стрелка с примитив, проверяващ дали е достигнато финалното състояние (State=Final) и преход към FINISH;
   b. Стрелка (примка) с примитив ReadWriteMove, извършващ всички основни обработки на Тюринг машината – четене на поредния символ от лентата, извличане на правило от структурата TF (функцията δ), прилагане на правилото – запис върху лентата Tape, промяна на текущото състояние State и препозициониране на главата Head.

ЗАКЛЮЧЕНИЕ

Въпреки, че и двата подхода за симулиране на стандартна Тюринг машина дават аналогичен резултат, облекченото дефиниране на УМ при императивния подход е свързано с много по-сериозни обработки, които трябва да се изпълняват вътре в примитивите. Декларативният подход е по-естествен и близък до човешкото мислене и деклариране на проблема. В [5] е разгледана и CNP/SPIDER реализация на много по-сложна недетерминирана Тюринг машина.

ЛИТЕРАТУРА

[1] K. Kratchanov, E.Golemanova, and T.Golemanov, “Control Network Programs and Their Execution,” in 8th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED 2009), Cambridge, UK, 2009, pp. 417–422.

[2] A. Turing, “On computable numbers,” Proc. London Math. Soc., vol. 42, pp. 230–265, 1936.

[3] P. Linz, An Introduction to Formal Languages and Automata, 5th edition. Jones & Bartlett, 2011.

[4] T. Golemanov, K.Kratchanov, and E.Golemanova, “SPIDER – A Language for Programming Through Control Networks,” in CompSysTech 2000, Sofia, Bulgaria, 2000, pp. 2091–2095.

[5] K. Kratchanov, E.Golemanova, T.Golemanov, and B. and Kulahcioglu, “Using Control Network Programming in Teaching Nondeterminism,” in CompSysTech 2012, Rousse, Bulgaria, 2012.

За контакти:

гл. ас. д-р Цанко Големанов, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, тел.: 082-888 681, e-mail: TGolemanov@uni-ruse.bg

Докладът е рецензиран.