Nr rejestracyjny: 2017/26/D/ST6/00201; Kierownik projektu: dr Lorenzo Clemente

# Zrozumieć rekursję

Lorenzo Clemente

Weryfikacja programów funkcyjnych jest dużym wyzwaniem w dziedzinie model checkingu. Cechy funkcyjne pojawiają się we wszystkich głównych językach programowania, takich jak na przykład C++, Java, Haskel, Python, Scala, Scheme i Erlang. Podstawą programowania funkcyjnego jest rekursja, jak widać w programie pełniącym rolę „hello world" programów funkcyjnych, tj. programie generującym ciąg Fibonacciego $0,1,1,2,3,5,8,\ldots$:

$$f(n+2) = f(n+1) + f(n), \text{ gdzie } f(0) = 0 \text{ i } f(1) = 1.$$

Jednakże gdy w rekursji umożliwiona jest manipulacja danymi (tutaj w formie operacji arytmetycznych '+' na liczbach naturalnych), natychmiastowo prowadzi to do nierozstrzygalności wszystkich nietrywialnych cech takich programów, tj. nie istnieje algorytm, który, dostawszy opis programu, zawsze terminuje i daje poprawną odpowiedź 'tak/nie' na rozważane pytanie. Poprzez usunięcie danych i unikanie interpretowania operacji arytmetycznych (tj. bez ich wykonywania), otrzymuje się *schemat rekursji*, który wiernie modeluje przepływ sterowania w programie.

$$F \to 0 \mid 1 \mid F + F. \tag{1}$$

W tym przypadku, otrzymujemy schemat rekursji, który wygląda tak samo jak klasyczna gramatyka bezkontekstowa. Jednakże, symbole takie jak "$F$" mogą same posiadać argumenty, więc w ogólnym przypadku, otrzymuje się schemat wyższego rzędu.

W tym projekcie proponujemy trzy problemy odnoszące się do schematów i weryfikacji. W pierwszym problemie, rozszerzamy schematy rekursji o znaczniki ograniczające dopuszczalny czas wykonania. W ten sposób, możemy wyrażać ograniczenia takie jak "wywołanie operacji F trwa 2 dwie jednostki czasu, a wywołanie operacji 0 i 1 traw jedną jednostkę". Przy tej konkretnej specyfikacji, wyrażenie "$0+0$" jest dozwolone ($1+1=2$ jednostki czasu) ale wyrażenie "$0+0+0$" nie jest dopuszczalne ($1+1+1=3$ jednostki czasu). Ta konkretna specyfikacja wyraźnie spełnia wszystkie wymagania, ale nie zawsze ma to miejsce. Ogólnie mówiąc, może być bardzo trudno stwierdzić, która sytuacja ma miejsce. Zaproponowaliśmy konkretny model zwany automatem czasowym ze stosem, który jest odpowiednikiem schematu czasowego pierwszego rzędu (jak nakreślono powyżej). Pokazaliśmy także, iż istnieje algorytm (działający w czasie podwójnie wykładniczym względem rozmiaru automatu), który może zadecydować o cechach tego modelu. W tym projekcie dążymy do przedstawienia bardziej wydajnego algorytmu (działającego w czasie pojedynczo wykładniczym) jak również wprowadzania bardziej wyrazistego języka służącego konstrukcji automatu. W drugim problemie odnosimy się do schematów wyższego rzędu jak zaprezentowano powyżej, które są uogólnieniem schematów rekursji poprzez rekursję wyższego rzędu, tj. gdzie symbole nieterminalne mogą posiadać argumenty funkcyjne. Patrząc na schemat jak w przypadku (1) powyżej, można by zapytać przykładowo, czy generuje on wszystkie wyrażenia postaci 0, 1, $0+0$, $0+1$, $0+(1+1)$,..., a bardziej ogólnie, wszystkie wyrażenia otrzymane ze stałych $0,1$ poprzez aplikacje binarnej operacji '+'. Podczas gdy pytanie to jest nierozstrzygalne dla schematów, problem ten może być częściowo ominięty poprzez rozważanie nad-aproksymacji schematu otrzymanej przez jego tak zwane *domknięcie w dół* (ang. downward closure). Intuicyjnie oznacza to, że każde wyrażenie wygenerowane przez schemat może także tworzyć podwyrażenia. Ostatnio pokazaliśmy jak obliczać domknięcia w dół dla wyrażeń liniowych (tj. wyrażeń bez symboli binarnych takich jak '+' powyżej) i planujemy rozszerzyć ten wynik do dowolnych wyrażeń. Wreszcie, w ostatnim problemie, zajmujemy się innym modelem, zwanym *sieciami Petriego* (ang. Petri nets), które są z grubsza nierekurencyjnymi programami z licznikami, oraz problemem zwanym *regularną separowalnością*. Podczas gdy sieci Petriego mogą być bardzo skomplikowane, w problemie regularnej separowalności szukamy prostych w opisie sposobów na rozróżnienie dwóch sieci. Jest to bardzo wymagający i trudny otwarty problem w tej dziedzinie.