Nr rejestracyjny: 2016/23/D/ST6/01387; Kierownik projektu: dr Filip Sieczkowski

Skalowalne metody wnioskowania o imperatywnych programach współbieżnych

Filip Sieczkowski

Każdy kto napisał kiedyś choć jeden program komputerowy wie jak łatwo jest popełnić w nim błąd, sprawiający że program nie będzie działał poprawnie. Nie wszyscy jednak muszą zdawać sobie sprawę że ten problem nie znika z rosnącym doświadczeniem programisty: jeśli używamy komputera do rozwiązania skomplikowanego problemu, jest niemal pewne że będzie trudno ustrzec się błędów w programie. Metody *testowania* oprogramowania są niezastąpione jeśli chcemy znaleźć błąd czy zlokalizować go w konkretnej części programu, jednak nie mogą dać nam absolutnej pewności program nie zawiera żadnych błędów. Uzyskanie takich gwarancji jest celem metod *weryfikacji* programów, dziedziny w której leży nasz projekt.

Jeden z fundamentalnych podziałów w zakresie weryfikacji to podział na metody *automatyczne*, analizujące kod programu bez konieczności interakcji z użytkownikiem, i metody w których podstawową rolę odgrywa człowiek *wspomagany* przez metody matematyczne czy systemy komputerowe. To rozgraniczenie wynika ze znanych ograniczeń komputerów: duża część interesujących własności programów jest zbyt skomplikowanych obliczeniowo, aby komputery mogły o nich wnioskować. W przypadku weryfikacji, powoduje to że metody automatyczne zwykle koncentrują się na stwierdzeniu nieobecności pewnych istotnych *klas błędów*, nie mówiąc jednak nic o *pełnej poprawności* programu — zaś metody wspomagane przez użytkownika pozwalają na precyzyjne wyrażenie *specyfikacji*, czyli oczekiwanego zachowania programu i, często wspomagane maszynowo, wykazanie że specyfikacja jest spełniona.

W ogólności, weryfikacja programów jest bardzo trudnym problemem, badanym od co najmniej pół wieku. Dopiero na przełomie wieków uzyskano jednak techniki, które dały nadzieję na skuteczne wnioskowanie o programach w realistycznych językach programowania. Kluczową cechą tych technik, znanych jako logiki separacyjne, jest *modularność* — cecha która sprawia że podobnie jak przy konstrukcji programów, możemy budować dowody większych programów z dowodów ich części składowych, *abstrahując* od większości szczegółów ich implementacji. To sprawia, że techniki te dają się zastosować do programów znacząco większych i bardziej skomplikowanych niż kiedykolwiek wcześniej rozważane.

Jednym z największych wyzwań podjętych w dziedzinie weryfikacji w ciągu ostatniej dekady jest znalezienie technik wnioskowania dla programów *współbieżnych*, w których wiele procesów komunikuje się ze sobą używając wspólnej pamięci do której wszystkie procesy mają dostęp. Jest to model obliczeń nieporównanie bardziej skomplikowany niż tradycyjne modele sekwencyjne, w których program był odizolowany od świata zewnętrznego — a jednocześnie wraz ze wzrostem liczby rdzeni w dzisiejszych procesorach, model który musi być coraz szerzej stosowany w praktyce.

W niniejszym projekcie podejmujemy zarówno kwestie pełnej poprawności programów współbieżnych, jak i ich automatycznej analizy, a także zadania leżące na pograniczu tych dziedzin. Po pierwsze, zajmujemy się problemem rozwoju logik separacyjnych dla programów współbieżnych: w tym zakresie chcemy doprowadzić do pewnego uproszczenia metod wnioskowania przy zachowaniu ich mocy wyrazu. Po drugie, chcemy zająć się metodami gwarantowania że współbieżny program jest *deterministyczny*, czyli że wynik programu jest jednoznacznie zależny od jego danych. To oznacza, w szczególności, że kolejność komunikacji procesów współbieżnych nie wpływa na wynik programu, co może znacząco uprościć wnioskowanie o działaniu takiego programu — a więc ułatwić programiście jego zrozumienie. Trzecie zadanie jakie sobie stawiamy jest szczególnie interesujące. Planujemy zbadać możliwość wykorzystania specyfikacji współbieżnych bibliotek i struktur danych przy automatycznej analizie kodu który jedynie *korzysta* z tych bibliotek. W praktyce możliwość takiego działania oznacza, że jeśli dowiedziemy pełną poprawność pewnych kluczowych, niewielkich części dużego programu współbieżnego, to będziemy mogli uzyskać istotne gwarancje dla całego programu przy pomocy automatycznych narzędzi. Wydaje się zatem, że techniki nad którymi będziemy pracować mogą w przyszłości dać programistom istotne gwarancje poprawności i uchronić użytkowników programów od nieprzyjemnych następstw błędów programistów.