Paweł Parys

rok urodzenia: 1983
e-mail: parys@mimuw.edu.pl
www: http://www.mimuw.edu.pl/∼parys/

Przebieg nauki i pracy zawodowej

- od 2009 — asystent na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego,
- X 2008 – I 2009 — staż naukowy w LaBRi, Bordeaux, Francja,
- od 2007 — studia doktoranckie z informatyki na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego,
- 2007 — magisterium z informatyki, praca pt. *Układy równań spełnione we wszystkich przemiennych półgrupach skończonych* pod kierunkiem dra hab. Mikołaja Bojańczyka, Uniwersytet Warszawski

Zainteresowania

- Dokumenty XML i ewaluacja zapytań XPath
- Automaty ze stosem wyższego rzędu
- Złożoność problemu wyliczania wyrażeń μ-rachunku w modelach skończonych
- Automaty czasowe

Dokumenty XML i ewaluacja zapytań XPath

Jednym z ważniejszych zagadnień związanych z dokumentami XML jest odnajdywanie w nich danych, które spełniają zadane kryteria. Często korzysta się przy tym z języka XPath, służącego do zapisywania zapytań. Jak wiadomo dokument XML możemy utożsamiać z pewnym drzewem, zwanym drzewem dokumentu. Zapytanie (unarne) XPath dla ustalonego dokumentu wyznacza pewien zbiór wierzchołków drzewa dokumentu. (Oprócz zapytań unarnych w XPath mamy także zapytania binarne, nazywane też wyrażeniami ścieżkowymi, które wyznaczają zbiór par wierzchołków. Podstawowy rodzaj zapytań to zapytania unarne; wyrażenia ścieżkowe mają raczej charakter pomocniczy, są używane wewnątrz zapytań unarnych.)

Rozważany przeze mnie problem algorytmiczny jest następujący: dane jest zapytanie XPath oraz dokument XML, należy znaleźć wszystkie wierzchołki drzewa dokumentu spełniające to zapytanie. Złożoność tego problemu istotnie zależy od tego, jakie zapytania są dozwolone, czyli jaki fragment języka XPath rozważamy. Uzyskane przeze mnie wyniki dotyczą fragmentu nazywanego FOXPath, w którym możemy porównywać wartości atrybutów między sobą (a więc jest to dosyć szeroki fragment). Dla tego fragmentu podałem algorytm liniowy ze względu na rozmiar dokumentu XML oraz sześcienny ze względu na rozmiar zapytania. Zwróćmy uwagę, że zależność od rozmiaru dokumentu jest znacznie ważniejsza niż zależność od rozmiaru zapytania, gdyż zazwyczaj dokumenty są znacznie większe niż zapytania. Wcześniej znane algorytmy dla tego fragmentu były przynajmniej kwadratowe ze względu na rozmiar dokumentu. Techniki użyte we wspomnianym algorytmie mają związek z teorią automatów. Warto zaznaczyć, że zapytania XPath nie można bezpośrednio przetłumaczyć na automat skończony dla drzew, gdyż język XPath jest zbyt bogaty.

Następnie wraz z współautorem zaproponowaliśmy model automatu, który:

- jest w stanie wyrazić dowolne zapytania FOXPath (a także więcej),
- w czasie liniowym ze względu na rozmiar dokumentu XML możemy obliczyć zbiór wierzchołków wybieranych przez ten automat (czyli zbiór wierzchołków wybierany przez zapytanie XPath przekształcone na ten automat).

Podejście to uogólnia nasze wcześniejsze wyniki dotyczące wyliczania zapytań XPath w czasie liniowym ze względu na rozmiar dokumentu. Jednocześnie podział algorytmu na dwa etapy (przekształcenie zapytania do automatu i wyliczenie automatu) powoduje, że staje się on bardziej zrozumiały.

Automaty ze stosem wyższego rzędu

Automaty ze stosem wyższego rzędu są naturalnym uogólnieniem automatów ze stosem: zamiast zwykłego stosu, w automacie drugiego rzędu mamy stos stosów, w automacie trzeciego rzędu stos stosów stosów, itd. Z najwyższego z tych stosów automat korzysta tak jak zazwyczaj (wkłada i usuwa symbole z wierzchołka stosu, widząc tylko najwyższy symbol). Automat drugiego rzędu może ponadto skopiować cały najwyższy stos i odłożyć go na stosie drugiego rzędu, a także usunąć najwyższy stos ze stosu drugiego rzędu. Analogicznie automat wyższych rzędów może postępować ze stosami wyższych rzędów.

Jak wiadomo, automaty skończone odpowiadają wyrażeniom regularnym (czyli, w pewnym sensie, programom bez rekurencji), natomiast automaty ze stosem odpowiadają gramatykom bezkontekstowym (czyli programom rekurencyjnym). Istnieje również analogiczna odpowiedniość między tzw. schematami (programami) rekurencyjnymi wyższych rzędów a automatami ze stosem wyższych rzędów, lecz nie jest ona doskonała. Okazuje się, że takie automaty ze stosem, jak opisano powyżej, odpowiadają tylko pewnej podklasie schematów rekurencyjnych, tak zwanym *bezpiecznym* schematom rekurencyjnym. Natomiast wszystkie schematy rekurencyjne odpowiadają pewnemu rozszerzeniu automatów ze stosem, wzbogaconych o dodatkową operację paniki (ang. *collapse*).

Do niedawna problemem otwartym było, czy klasy te są istotnie różne, czyli czy automat wzbogacony o operację paniki może rozpoznać więcej języków niż automat bez paniki. Udowodniłem, iż klasy te rzeczywiście są różne. Konkretnie pokazałem, iż istnieje język rozpoznawany przez automat drugiego rzędu z paniką, który nie jest rozpoznawany przez automat dowolnie wysokiego rzędu bez paniki.

Przy okazji rozwinąłem pewne narzędzia ułatwiające dowodzenie, że pewien język nie jest rozpoznawany przez automat ze stosem wyższego rzędu; w szczególności podałem lemat o pompowaniu.

Złożoność problemu wyliczania wyrażeń μ-rachunku w modelach skończonych

Przypuśćmy, że chcemy obliczać wyrażenia μ-rachunku w określonej kracie skończonej (w naszym przypadku będzie to $\{0,1\}^n$). Znanym problem otwartym jest pytanie, czy można to robić w czasie wielomianowym; jest to równoważne znajdowaniu strategii wygrywającej w grze parzystości i paru innym ważnym problemom. Ogólna postać wyrażenia μ-rachunku to najpierw występujące na zmianę operatory μ i ν, a wewnątrz pewna funkcja monotoniczna F. Wiele zależy od tego, jak funkcja F jest dana. Typowo zakłada się, że F dana jest za pomocą formuły logicznej. Ja badałem nieco inny model, gdzie F dana jest jako czarna skrzynka; jedyny sposób, w jaki możemy ją oglądać, to obliczać jej wartości dla danych argumentów. Uzyskałem dwa wyniki dotyczące złożoności takiego problemu.

Pierwszy wynik dotyczy ograniczenia dolnego. Ograniczyłem się do wyrażeń, gdzie występuje tylko jeden operator μ i jeden ν. Udowodniłem, że wówczas potrzeba zadać prawie n^2 pytań do funkcji F, aby obliczyć wartość wyrażenia. Może to być pierwszy krok w kierunku udowodnienia, że w ogólności (gdy jest więcej operatorów μ i ν) potrzeba wykładniczo wiele pytań (tego nie udało mi się dowieść). Wynik mój pokazuje również, że używanie na raz operatorów μ i ν daje wyrażenia trudniejsze do obliczania niż takie, które zawierają tylko μ lub tylko ν.

Drugi wynik dotyczy ograniczenia górnego. Zwykle przyjmuje się, że wejściem jest zarówno formuła μ-rachunku, jak i rozmiar kraty n. Ja przeciwnie, przyjmuję rozmiar kraty n jako stały. Pokazałem, że wówczas wartość wyrażenia można obliczać w czasie wielomianowym względem rozmiaru wyrażenia (o ile w wyrażeniu najpierw występują na zmianę operatory μ i ν, a wewnątrz jest pewna funkcja monotoniczna F; każde inne wyrażenie można sprowadzić do wyrażenia tej postaci, jednak niestety trzeba wówczas zwiększyć rozmiar kraty). Wynik ten był już znany, gdy funkcja F dana jest za pomocą formuły logicznej, jak zwykle się przyjmuje. Ja udowodniłem wynik ogólniejszy, gdyż jedyne czego potrzebuję od funkcji F, to możliwości jej obliczania (F dana jest więc za pomocą procedury).

Automaty czasowe

Automaty czasowe to znane rozszerzenie automatów skończonych używane przy weryfikacji systemów odwołujących się do zegara. Zajmowałem się problemem pustości takiego automatu, czyli rozstrzyganiem czy dany automat akceptuje jakiekolwiek słowo. W zależności od tego, jaką konkretnie rozważa się klasę automatów, problem ten jest rozstrzygalny lub

nie. Wyniki tego rodzaju były znane dla wielu naturalnych klas automatów. Ja zajmowałem się słabymi automatami alternującymi z jednym zegarem, dla słów nieskończonych. Chciałem ustalić przy jakich założeniach pustość takich automatów jest rozstrzygalna. Wraz z współautorem udowodniliśmy, że problem jest rozstrzygalny, gdy automat ma tylko dwa poziomy: najpierw stany egzystencjalne, potem uniwersalne. W każdym innym przypadku problem jest nierozstrzygalny. Ponadto problem pozostaje nierozstrzygalny, gdy ograniczamy się do automatów nie sprawdzających równości ze stałą (jedynie nierówności).

Publikacje

[1] Mikołaj Bojańczyk, Paweł Parys. XPath evaluation in linear time. *Journal of the ACM*, 58:17:1–17:33, July 2011.

[2] Paweł Parys. Collapse operation increases expressive power of deterministic higher order pushdown automata. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science*, pages 603–614, 2011.

[3] Mikołaj Bojańczyk, Paweł Parys. Efficient evaluation of nondeterministic automata using factorization forests. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, pages 515–526, 2010.

[4] Paweł Parys, Igor Walukiewicz. Weak alternating timed automata. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 273–284, 2009.

[5] Paweł Parys. XPath evaluation in linear time with polynomial combined complexity. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 55–64, 2009.

[6] Paweł Parys. Systems of equations satisfied in all commutative finite semigroups. In *Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structures*, pages 261–272, 2008.

[7] Mikołaj Bojańczyk, Paweł Parys. XPath evaluation in linear time. In *Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 241–250, 2008.

[8] Krzysztof Onak, Paweł Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 379–388, 2006.

[9] Paweł Parys. Some results on complexity of mu-calculus evaluation in the black-box model. W recenzji.

[10] Paweł Parys, Igor Walukiewicz. Weak alternating timed automata. W recenzji.