Artur Jeż

Curriculum vitae

Data urodzenia:		4 V 1982
Miejsce urodzenia:		Wrocław

Obecnie:	od 2012	postdoc Max Planck Institut fuer Informatik
	od 2010	adiunkt Instytut Informatyki Wydział Matematyki i Informatyki Uniwersytet Wrocławski
	(od 2012	urlop naukowy)
Email:		aje@cs.uni.wroc.pl
Strona domowa:		http://www.ii.uni.wroc.pl/~aje/

Poprzednio:	2006–10	doktorant Studium Doktoranckie Informatyki Wydział Matematyki i Informatyki Uniwersytet Wrocławski

---

Wykształcenie:	2006	Magister informatyki, magister matematyki Uniwersytet Wrocławski
	2010	Doktor nauk matematycznych — informatyka Uniwersytet Wrocławski

---

Osiągnięcia:	2002–2005	Stypendium za wyniki w nauce (trzykrotnie) Ministerstwo Edukacji Narodowej i Sportu
	2006	Nagroda im. J. Marcinkiewicza (III stopnia) za najlepszą pracę magisterską z matematyki
	2007	Best paper in Theory Track, CSR
	2009	Best Student Paper, MFCS
		Stypendium im. Maxa Borna
		Stypendium Samorządu Wrocławia
	2010	Stypendium START Fundacja na rzecz Nauki Polskiej
		Doktorat obroniony z wyróżnieniem
	2011	Nagroda Prezesa Rady Ministrów za rozprawę doktorską
	2012	Wyróznienie: Nagroda im. Witolda Lipskiego

Przebieg pracy naukowej

W swojej pracy naukowej zajmowałem się głównie problematyką języków formalnych oraz ich związkami z innymi dziedzinami informatyki. Poniżej przedstawiam główne obszary swoich zainteresowań, kolejność odzwierciedla moją ocenę ważności uzyskanych wyników.

Kompresja gramatykowa. Moje obecne badania skupiają się wokół kompresji przy użyciu tzw. SLP (Straight-Line Programs), są one po prostu gramatykami bezkontekstowymi generującymi dokładnie jedno słowo. SLP są popularnym modelem kompresji, gdyż z jednej strony mają prostą i matematycznie elegancką definicję, z drugiej zaś dość dobrze oddają standard kompresji LZ (plik skompresowany w formacie LZ można przekształcić do SLP logarytmicznie większego).

W swojej pracy zaproponowałem ogólną metodę *lokalnej rekompresji*: polega ona na budowaniu „od dołu" kanonicznych SLP dla słów występujących w instancji. Budowanie to polega na iterowaniu operacji zastępowania pary symboli ab (gdzie zarówno a jak i b mogą być literami lub nieterminalami) występującej w instancji przez nowy nieterminal. Aby operacja to była możliwa, czasami potrzebna też jest lokalna zmiana zmiennych/nieterminali w instancji.

Równania słów. Problem spełnialności równań słów są jednym z najstarszych i najbardziej intrygujących problemów w teorii informatyki. Jego pierwsze rozwiązanie zostało przedstawione przez G. Makanina. Algorytm Makanina był wielokrotnie ulepszany i analizowany, jednak najlepszy obecnie algorytm (działający w PSPACE) sprawdzający spełnialność równań w słowach został zaproponowany przez W. Plandowskiego, który użył zupełnie innego podejścia, niż Makanin. Poza samą spełnialnością, badania równań słów skupiały się również na konstrukcji algorytmów generujących rozwiązania (tu również najlepszy obecnie algorytm został zaproponowany przez W. Plandowskiego) jak również na dowodzeniu licznych własności rozwiązań tych równań. Cechą wspólną większości algorytmów jak również wyników czysto teoretycznych jest to, że są trudne, tzn. zarówno konstrukcje, jak i dowody własności są zwykle dość skomplikowane.

Używając metody lokalnej rekompresji udało mi się zaprezentować nowe, stosunkowe proste dowody większości faktów znanych dla równań w słowach [19], tj. podałem algorytm działający w pamięci wielomianowej, który sprawdza spełnialność równania słów. Jednocześnie, jest on w stanie wygenerować wszystkie rozwiązania tego równania. Analiza tego algorytmu pokazuje, iż rozwiązanie minimalnej długości jest najwyżej podwójnie wykładnicze. Analiza innych własności tego algorytmu udowadnia, że tzw. *wykładnik periodyczności* rozwiązania jest najwyżej wykładniczy w stosunku do rozmiaru wejścia.

Podany algorytm jest prosty, zarówno sformułowanie oraz analiza nie wymagają znajomości zaawansowanych własności rozwiązań czy też twierdzeń kombinatoryki słów, co odróżnia go od większości algorytmów i twierdzeń w dziedzinie.

Problem rozpoznawania skompresowanego słowa przez skompresowany automat. Współcześnie, coraz większa ilość danych przechowywana i przekazywana jest w skompresowanej postaci. W związku z tym należy być przygotowanym, iż słowa pojawiające się jako wejście będą zadane w skompresowanej postaci. Można oczywiście takie słowo zdekompresować, lecz niweczy to większość zysku wynikającego z kompresji.

W przypadku języków formalnych, najważniejsze problemy związane ze słowami, dotyczą rozpoznawania/generowania przez rozmaite formalizmy (gramatyki, automaty, wyrażenia itp.). Jednym z problemów, którego złożoność obliczeniowa nie była znana, był problem rozpoznawania skompresowanego słowa przez skompresowany automat, tj. dla danego SLP oraz automatu N, mamy odpowiedzieć, czy N rozpoznaje słowo zadane przez to SLP; dodatkowo, N ma przejścia etykietowane słowami, które również zadane są jako SLP. Wiadomo było, iż problem ten jest w PSPACE oraz że jest NP-trudny. Pokazałem, że jest on w NP, oraz w P, jeśli dodatkowo N jest deterministyczny.

Skompresowane rozpoznawanie wzorca. Z algorytmicznego punktu widzenia najważniejszą i najczęstszą operacją, którą wykonuje się na skompresowanym tekście, jest wyszukiwanie wzorca — dzięki temu możemy sprawdzić, czy interesujące nas słowa kluczowe pojawiają się w tekście; jeśli tak, to należy go zdekompresować w celu dalszej obróbki.

Algorytmy dla tego problemu różnią się oczywiście w zależności od użytej kompresji, zajmowałem się wersją problemu, w której zarówno wzorzec jak i tekst skompresowane są przy użyciu SLP. Jak wcześniej wspomniałem, SLP dobrze modeluje standard kompresji LZ, tym samym problem nie jest zupełnie niepraktyczny.

Poprzednio znany był algorytm o czasie działania $\mathcal{O}(m^2n)$, gdzie m jest rozmiarem skompresowanego wzorca, zaś n rozmiarem skompresowanego tekstu. Używając metody rekompresji, podałem algorytm o czasie działania $\mathcal{O}((m+n)\log M \log(n+m))$, ponieważ $M \leq 2^m$, podany algorytm jest istotnie szybszy niż poprzednio znany.

Układy równań nad zbiorami liczb. Początkowo w swojej pracy naukowej zajmowałem się układami równań nad zbiorami liczb naturalnych. Dość wcześnie w teorii języków formalnych zauważono, iż narzędzia definiujące języki formalne mogą być użyte również do definiowanie zbiorów liczb naturalnych: w najprostszym przypadku alfabetu jednoliterowego, słowo a^n można zidentyfikować z liczbą n. W szczególności, operacja konkatenacji odpowiada dodawaniu.

Najpopularniejszymi formalizmami, służącymi do definiowania języków formalnych, są gramatyki i automaty. Trzecim, zyskującym ostatnio popularność, są układy równań języków formalnych. Z jednej stron, mają one sporą mocą wyrazu, z drugiej, ich semantyka jest jednocześnie prosta i dobrze matematycznie określone, może też być łatwo zaadaptowana do konkretnych zastosowań.

W przypadku alfabetu jednoliterowego układy takie można w naturalny sposób interpretować jako układy równań nad zbiorami liczb naturalnych, tj.

$$\varphi_i(X_1, \ldots, X_n) = \psi_i(X_1, \ldots, X_n), \text{ dla } i = 1, \ldots, m,$$

w których zmienne reprezentują podzbiory liczb naturalnych (całkowitych), a wyrażenia φ_i, ψ_i używają operacji boolowskich oraz *dodawania*, zdefiniowanego jako

$$(*) \qquad A + B = \{a + b \,:\, a \in A,\, b \in B\}.$$

Podałem klasyfikację rozwiązań takich układów, w zależności od dozwolonych operacji, postaci równań oraz tego, czy rozpatruje się je nad zbiorami liczb naturalnych czy też całkowitych.

Ważną podklasą układów równań języków formalnych są układy w *postaci rozwiązanej*, tj.

$$X_i = \varphi_i(X_1, \ldots, X_n), \text{ dla } i = 1, \ldots, n.$$

Takie równań odpowiadają gramatykom formalnym, np. jeśli φ_i mogą używać tylko operacji sumy oraz konkatenacji, to odpowiadają one gramatykom bezkontekstowym (CFG). W szczególności, pozwala to na zadanie naturalnej semantyki dla rozszerzenia CFG o przecięcie, które może występować w ciele dowolnej reguły; otrzymane w ten sposób gramatyki znane są jako *gramatyki koniunkcyjne*, które są najbardziej naturalnym i obiecującym rozszerzeniem gramatyk bezkontekstowych o operację przecięcia.

Wspólnie z A. Okhotinem pokazaliśmy, że:

- układy równań w postaci rozwiązanej używające operacji $\cap$, $\cup$, $+$ definiują (poprzez rozwiązania jedyne) wszystkie zbiory liczb naturalnych, których notacja k-pozycyjna dla pewnego k jest rozpoznawana przez automat kratowy [2];
- problem przynależności liczby do najmniejszego rozwiązania układu równań w postaci rozwiązanej używającego operacji $\cap$, $\cup$, $+$ jest EXPTIME-zupełny [4], problem ten zachowuje swoją złożoność nawet wtedy, gdy rozważamy pojedyncze równanie z jedną zmienną [9].

W przypadku równań w postaci ogólnej, wiadomo, że układy równań języków formalnych są obliczeniowo zupełne: dla alfabetu zawierającego przynajmniej dwie litery klasa rozwiązań jedynych (największy, najmniejszych) pokrywa się z klasę języków rekurencyjnych (rekurencyjnie przeliczalnych (r.e.), ko-rekurencyjnie-przeliczalnych (ko-r.e.)). Udało się nam uzyskać analogiczny wynik dla układów równań nad zbiorami liczb naturalnych: klasa rozwiązań jedynych (najmniejszych, największych) dla układów równań nad zbiorami liczb naturalnych używających operacji $+$, $\cup$ (lub $+$, $\cap$) oraz jednoelementowych stałych stanowi klasę zbiorów rekurencyjnych (r.e., ko-r.e.) [5].

Wynik ten wzmocniliśmy: jeśli rozważyć równania używające jedynie operacji $+$, to klasa rozwiązań jedynych (najmniejszych, największych) koduje wszystkie zbiory rekurencyjne (r.e., ko-r.e.) [7]. Analogiczne wyniki zachodzą również, jeśli rozważamy pojedyncze równanie używające operacji $\cap$, $\cup$ i $+$ oraz tylko jednej zmiennej [14].

Naturalną kontynuacją tych badań było rozważanie równań nad zbiorami liczb całkowitych. Nietrywialne ograniczenie górne, które sprawia, że wszystkie rozwiązanie jedyne (najmniejsze, największe) w przypadku zbiorów liczb naturalnych są rekurencyjne (r.e., ko-r.e.) nie zachodzi w przypadku liczb całkowitych. Tym niemniej istnieje stosunkowo proste ograniczenie górne: układy równań można przetłumaczyć na formuły logiki używające zbiorów liczb naturalnych odpowiadających zmiennym. Dzięki prostej kwantyfikacji otrzymujemy, że rozwiązania najmniejsze (największe) są określane przy użyciu uniwersalnych (egzystencjalnych) formuł drugiego rzędu arytmetyki Peano. Tym samym rozwiązania najmniejsze (największe) składają się zawsze ze zbiorów Π^1_1 (Σ^1_1), a rozwiązania jedyne są więc zbiorami z klasy Δ^1_1.

W przypadku równań w postaci rozwiązanej, jeśli dopuścimy tylko jedną operację boolowską, uzyskane rozwiązania są trywialne, badaliśmy więc układy, które używają operacji $\cup$, $\cap$ oraz $+$. Pokazaliśmy, że klasa rozwiązań największych to klasa zbiorów Σ^1_1 [13]; to ograniczenie dolne pokrywa się z trywialnym ograniczeniem górnym. Z drugiej strony, pokazaliśmy, że klasa rozwiązań najmniejszych takich układów równań to dokładnie klasa zbiorów r.e. [13].

W przypadku układów równań w postaci ogólnej, pokazaliśmy, że ograniczenie górne na klasę rozwiązań jedynych (Δ_1^1) jest osiągane już w przypadku układów równań układów używających jedynie operacji $\cup$ oraz $+$ [11].

Minimalizacja (z błędami) automatów. Jednym z podstawowych problemów w teorii automatów skończonych jest minimalizacja, tj. dla ustalonego automatu konstrukcja minimalnego automatu rozpoznającego ten sam język. Warunek rozpoznawania tego samego języka można osłabić, zezwalając, aby język różnił się od wejściowego w niewielkim stopniu. W literaturze znane jest np. pojęcie minimalnego automatu k-*pokrywającego*: dla danego automatu chcemy skonstruować minimalny automat rozpoznający dokładnie te same słowa długości nie większej niż k. Niedawno pojawiły się też podejścia dualne do automatów pokrywających: tzw. *hiper-minimalizacja* oraz k-*minimalizacja*. W pierwszej pozwalamy, aby skonstruowany automat różnił się od wejściowego na skończenie wielu słowach, w drugim, aby różnił się na słowach długości nie większej niż k. Intuicyjnie odpowiada to stwierdzeniu, że interesujemy się jedynie granicznym zachowaniem automatu.

Dla problemu konstrukcji minimalnego automatu k-pokrywającego znany jest algorytm o czasie działania $\mathcal{O}(n \log n)$. Wspólnie Andreasem Malettim podaliśmy dużo prostszy algorytm o tym samym czasie działania, ponadto nasz algorytm może być wykorzystywany do jednoczesnego obliczania automatów k-pokrywających dla różnych wartości k [16].

Wspólnie z Pawłem Gawrychowskim (a następnie też Andreasem Malettim) zajmowałem się też pozostałymi wymienionymi problemami, tj. konstrukcją hiper-minimalizacją i k-minimalizacją. Podaliśmy algorytm o czasie działania $\mathcal{O}(n \log^2 n)$ [10] dla problemu k-minimalizacji, który następnie poważnie uprościliśmy i poprawiliśmy [15] (w nowszej wersji również może on być użyty do równoległego obliczania automatów k-minimalnych dla różnych wartości k). Poprzednio znany był jedynie kwadratowy algorytm hiper-minimalizujący,

Następnie zajęliśmy się wariantami wyżej wymienionych problemów: innym podejściem do poprawienia pojęcia hiper-minimalizacji jest *dokładna hiper-minimalizacja*, tj. konstrukcja hiper-minimalnego automatu, którego język dodatkowo różni się od wejściowego na najmniejszej możliwej liczbie słów. Andreas Maletti pokazał, że ten problem można rozwiązać w czasie $\mathcal{O}(n^2)$. Wspólnie z nim pokazaliśmy, że analogiczny problem *dokładnej* k-*minimalizacji* jest NP-trudny [15]. Redukcję, użytą do pokazania trudności tego problemu można zaadaptować do zbliżonego problemu konstrukcji minimalnego automatu rozpoznającego język różniący się od wejściowego na najwyżej k słowach. Tym samym pokazaliśmy, że ten problem również jest NP-trudny [15].

Ponadto, wspólnie z Andreasem Malettim, rozszerzyliśmy pojęcie hiper-równoważności na automaty rozpoznające drzewa oraz podaliśmy algorytm hiperminimalizujący dla takich automatów [20], jego czas działania to $\mathcal{O}(\ell m n)$, gdzie ℓ jest maksymalnym stopniem drzewa, m jest ilością różnych przejść automatu a n liczbą stanów.

Publikacje

[1] A. Jeż, „Conjunctive grammars generate non-regular languages over unary alphabet", *Development in Language Theory (DLT)*, (Turku, Finlandia 2007), LNCS 4588, 242–253,
pełna wersja: *International Journal of Foundations of Computer Science* 19:3 597–615 (2008), wydanie spcjalne po DLT 2007.

[2] A. Jeż, A. Okhotin, „Conjunctive grammars over a unary alphabet: undecidability and unbounded growth", *International Computer Science Symposium in Russia (CSR)*, (Jekaterynburg, Rosja 2007), LNCS 4649, 168–181,
pełna wersja: *Theory of Computing Systems* 46:1 27–58 (2010), wydanie specjalne: CSR 2007.

[3] M. Grech, A. Jeż, A. Kisielewicz, „Graphical complexity of products of permutation groups", *Discrete Mathematics*, 308:7 1142–1152 (2008).

[4] A. Jeż, A. Okhotin, „Complexity of solutions of equations over sets of natural numbers", *Symposium on Theoretical Aspects of Computer Science (STACS)*, (Bordeaux, Francja 2008), LIPIcs 1, 373–384,
pełna wersja: *Theory of Computing Systems* 48:2, 319–342 (2011).

[5] A. Jeż, A. Okhotin, „On the computational completeness of equations over sets of natural numbers", *International Colloquium on Automata, Languages and Programming (ICALP)*, (Reykjavik, Islandia 2008), LNCS 5126, 63–74.

[6] M. Bieńkowski, M. Chrobak, C. Dürr, M. Hurand, A. Jeż, Ł. Jeż, G. Stachowiak, „Collecting Weighted Items from a Dynamic Queue", *Symposium on Discrete Algorithms (SODA)*, (Nowy Jork, USA 2009) SIAM SODA, 1126–1135
pełna wersja: przyjęta do *Algorithmica*.

[7] A. Jeż, A. Okhotin, „Equations over sets of natural numbers with addition only", *Symposium on Theoretical Aspects of Computer Science (STACS)*, (Fryburg, Niemcy 2009), LIPIcs 3, 577–588.

[8] A. Jeż, J. Łopuszański, „On the Two-Dimensional Cow Search Problem", *Information Processing Letters* 109:11 543–547 (2009).

[9] A. Jeż, A. Okhotin, „One-nonterminal conjunctive grammars over a unary alphabet", *International Computer Science Symposium in Russia (CSR)*, (Nowosybirsk, Rosja 2009), LNCS 5675, 191–202
pełna wersja: *Theory of Computing Systems* 49:2, 319–342 (2011), wydanie specjalne: CSR 2009.

[10] P. Gawrychowski, A. Jeż, „Hyper-minimisation made efficient", *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, (Nowy Smokowiec, Słowacja 2009) LNCS 5734, 356–368.

[11] A. Jeż, A. Okhotin, „On equations over sets of integers", *Symposium on Theoretical Aspects of Computer Science (STACS)*, (Nancy, Francja 2010), LIPIcs 5, 477–488,
pełna wersja: przyjęta do *Theory of Computing Systems*, wydanie specjalne: STACS 2010.

[12] P. Gawrychowski, A. Jeż, Ł. Jeż, „Validating the Knuth-Morris-Pratt failure function, fast and online", *International Computer Science Symposium in Russia (CSR)*, (Kazań, Rosja 2010), LNCS 6072, 132–143,
pełna wersja: zgłoszona do *Theory of Computing Systems*, wydanie specjalne: CSR 2010.

[13] A. Jeż, A. Okhotin, „Least and greatest solutions of equations over sets of integers", *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, (Brno, Czechy 2010), LNCS 6281, 441–452.

[14] A. Jeż, A. Okhotin „Univariate Equations Over Sets of Natural Numbers" *Fundamenta Informaticae* 104, 329–348 (2010)

[15] P. Gawrychowski, A. Jeż, A. Maletti „On Minimising Automata with Errors" *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, (Warszawa, 2011) LNCS 328–388.

[16] A. Jeż, A. Maletti „Computing all ℓ-cover automata fast" *„International Conference on Implementation and Application of Automata (CIAA)*, (Blois, Francja 2011) LNCS 6807, 203–214.

[17] Artur Jeż, „Compressed Membership for NFA (DFA) with Compressed Labels is in NP (P)" *Symposium on Theoretical Aspects of Computer Science (STACS)*, (Paryż, Francja 2012), LIPICs 14, 136–147
pełna wersja: zgłoszona do *Theory of Computing Systems*, wydanie specjalne: STACS 2012.

[18] „Faster fully compressed pattern matching by recompression", *International Colloquium on Automata, Languages and Programming (ICALP (A))* (Warwick, Wielka Brytania 2012), LNCS 7391, 533–544.

[19] Artur Jeż „Recompression: a simple and powerful technique for word equations", *CoRR 1203.3705, 2012*, *zgłoszone*.

[20] A. Jeż, A. Maletti „Hyper-Minimization for Deterministic Tree Automata" *„International Conference on Implementation and Application of Automata (CIAA)*, (Porto, Portugalia 2012), LNCS 7381, 217–228.

[21] A. Jeż, A. Okhotin, „On the number of nonterminal symbols in unambiguous conjunctive grammars", *International Workshop on Descriptional Complexity of Formal Systems (DCFS)*, (Braga, Portugalia 2012), LNCS 7386, 183–195.